MASTER EN SCIENCES INFORMATIQUES

Un outil ergonomique d'aide à la réalisation d'horaires

Bodden, Martine

Award date:
1989

Awarding institution:
Universite de Namur

UN OUTIL ERGONOMIQUE d' AIDE à la REALISATION d' HORAIRES.

réalisé par BODDEN Martine

pour l' obtention du titre de LICENCIE et MAITRE en INFORMATIQUE

année académique 1988 - 1989

Je remercie Mademoiselle Chandelon et Monsieur Leclercq pour leurs conseils judicieux ainsi que pour la disponibilité dont ils ont fait preuve.

Je remercie l' Institut d' Informatique des Facultés Notre Dame de la Paix à Namur pour le matériel et le logiciel qui ont été mis à ma disposition.

Je remercie également toutes les personnes qui de près ou de loin m' ont aidée dans la réalisation de ce travail.

Table des Matières.

INTRODUCTION

1.1.Pourquoi une Aide ?

La conception de l' horaire des cours d' un établissement de l' enseignement secondaire est depuis longtemps considérée comme un casse-tête par les responsables administratifs d' un tel établissement.

Chaque année, ce travail préoccupe les responsables administratifs de nombreuses écoles, mais de bons horaires sont rarement trouvés.

L' apparition de l' informatique comme outil de gestion a fait naître le désir d' utiliser l' ordinateur comme **soutien** lors de l' établissement d' un horaire.

Les horaires conçus manuellement présentent des inconvénients dus entre autres au fait que l' ampleur du problème combinatoire dépasse les capacités d' appréhension globale d' un individu.

Il en résulte des oublis, des incompatibilités, des difficultés de modification, une dépense de temps et d' énergie considérable ainsi que des difficultés liées à un **manque d' information** (composition des cours, indisponibilités, ...).

De nombreuses conversations avec les responsables des horaires ont fourni une vue globale et assez complète du problème.

A côté des **données de base** (lieux, acteurs, activités), on distingue de nombreuses **contraintes** (ne pas placer deux cours différents du même professeur à une même heure, indisponibilité de locaux, ...) de nature pratique ou pédagogique.

Le but de ce travail est dès lors d' établir un logiciel d' aide à la manutention d' un horaire et cela d' un point de vue purement **ergonomique**.

1.2.Quelle Aide ?

Le but de ce travail ne sera pas de réaliser un logiciel de conception de ces horaires (de nombreux travaux ont déjà été effectués sur ce sujet) mais principalement d' établir un logiciel d' aide à la manutention et à la réalisation d' un horaire de cours pour l' enseignement secondaire.

Ce travail portera principalement sur l' aspect ergonomique d' utilisation.

En effet, ce programme sera utilisé par des non initiés; ce qui demande - soit de la part de l' utilisateur une grande motivation et éventuellement de réelles connaissances de son environnement,

- soit de la part du concepteur une prise de conscience de la difficulté que rencontreront "ses" futurs utilisateurs face au produit et une volonté de rendre celui-ci le plus courant possible.

C' est cette dernière solution qui vous sera proposée. L'aide à la manutention d' horaires se fera en 3 phases:

- pré-contrôle
  l' utilisateur souhaite connaître la disponibilité de certaines activités, de certains acteurs, ...

- post-contrôle
  une fois que l' utilisateur aura proposé un horaire, le logiciel aura pour mission de vérifier qu' il n' y a aucune incohérence quant aux contraintes préalablement définies.

- contrôle de fin
  le logiciel devra tester l' achèvement de l' horaire.

1.3.Comment Aider ?

Pour réaliser un outil de gestion présentant une facilité d' utilisation, un certain nombre de concepts et d' outils sont mis à sa disposition.

C' est le cas par exemple des menus déroulants limitant les erreurs dues au dialogue trop imprécis avec l' ordinateur.

La réalisation d' un horaire demande également un important dispositif de stockage de données ainsi qu' un dispositif d' accès(il suffit de penser au nombre d' étudiants dans une école).

Pour ce dispositif, nous utiliserons une Base de Données(1).

---

(1) NDBS : B.D. mise grâcieusement à la disposition des étudiants par l' Institut d' Informatique des Facultés Notre Dame de la Paix

ASPECTS ERGONOMIQUES

2.1.Introduction. [2.1.]

2.1.1.Généralités

Définition.

L' ergonomie est une discipline scientifique dont l' objet fondamental est d' étudier l' humain dans ses rapports fonctionnels avec le travail.

L' élément humain étant le déterminant principal du processus de travail, l' ergonomie utilise donc des données physiques, physiologiques, psychologiques, sociales et culturelles pour définir les normes de travail en fonction des individus, des tâches à accomplir, des outils utilisés et de l' environnement, le tout dans une approche intégrée.

L' ergonomie tient compte des quatre éléments suivants:

- les individus: les travailleurs dans leur milieu de travail,
- les tâches: ce que les travailleurs font et la manière dont ils accomplissent leur tâche,
- les outils: les technologies, les équipements, les mécanismes et tout autre apport utilisé dans l' accomplissement de leur tâche,
- l' environnement: tous les aspects physiques du milieu de travail et du poste de travail.

Objectifs de l' Ergonomie.

L' ergonomie vise essentiellement à procurer un environnement de travail sain, sécuritaire, confortable, facile à utiliser et favorisant la performance ou la productivité et ce, de concertation avec la tâche ou les outils, avec les technologies ou les équipements requis.

2.1.2.Démarches Ergonomiques.

Devant un problème donné, les démarches ergonomiques peuvent être multiples:

- L' **analyse du comportement** du travailleur face à une tâche donnée permet d' optimiser la performance du travailleur, soit par une formation adéquate, soit par des rapports de toutes sortes pour accomplir le travail.
- L' **analyse de la tâche** permet d' évaluer ce qui est nécessaire à chacun des éléments du poste de travail pour accomplir la tâche demandée.
  Cela permet aussi d' identifier tout ce qui peut entraver la bonne marche du travail.

- L' **analyse des outils** étudie la relation entre l' humain et la tâche à accomplir, afin de recueillir des données qui sont utilisées dans la fabrication des outils nécessaires à la tâche visée et aussi à la formation du travailleur pour optimiser l' interface homme-machines.

- L' **analyse de l' environnement** se penche sur tous les facteurs physiques pouvant influencer la performance du travailleur: luminosité, qualité de l' air, acoustique, espace, câblage, ...
  Ces différentes exigences reliées à l' individu, à la tâche à accomplir, à la technologie utilisée vont façonner la planification du poste ou de l' aire de travail.

2.1.3.Type d' Ergonomie.

Il s' agit ici de souligner certains points qui conditionnent la qualité de vie au travail, et de ce fait, l' utilisation adéquate et plus aisée du matériel bureautique.

Les principaux types d' ergonomie sont:

- Ergonomie visuelle,

- Ergonomie auditive,

- Ergonomie posturale,

- Ergonomie psycho-sociale,

- Ergonomie du logiciel.

Avec l' apparition des micro-ordinateurs, le logiciel doit particulièrement s' adapter:

- aux mécanismes de la communication,
- aux personnes qui en sont les utilisateurs,
- à l' environnement dans lequel il se déroule.

Plusieurs facteurs déterminent l' ergonomie du logiciel centré sur les utilisateurs des programmes:

- la visualisation sur écran,
- les langages de commande.

L' ergonomie des logiciels vise à augmenter l' aspect convivial du dialogue et de l' usage du logiciel.

2.2.L' Ergonomie du Logiciel. [2.2.] [2.6.]

2.2.1.Ergonomie: Réponse à Tous les Problèmes ?

Il est important de constater que la conception d' un logiciel adapté aux utilisateurs n' est pas en mesure de résoudre l' ensemble des problèmes humains posés par l' introduction de l' informatique mais qu' inversement, il ne peut pas y avoir un processus d' informatisation réussi sans optimisation de la manière dont l' homme est amené à communiquer avec l' ordinateur.

2.2.2.Différents Points de Vue.

Des personnes différentes se trouvent sur le parcours du développement de logiciels interactifs (ex: ergonome, informaticien, utilisateur, conseiller en organisation, ...).

Nous ne retiendrons que les trois premiers.

L' objectif de ces personnes est différent: en effet leurs perceptions de l' interaction homme-machine diffèrent.

Décrivons les différents points de vue:

UTILISATEUR:

Il souhaite disposer d' un logiciel lui permettant d' exécuter sa tâche avec le plus de facilité possible.

Les paramètres pour une découpe "utilisateur" sont:

U1- lisibilité des sorties,
U2- adaptation du logiciel aux situations réelles de travail,
U3- aide à l' utilisation et à l' apprentissage,
U4- opérations redondantes,
U5- initiatives possibles pour l' utilisateur.

ERGONOME:

Il souhaite l' amélioration de la communication homme-machine, il observe le couple homme-machine de l' extérieur.

Les paramètres pour une découpe "ergonome":

E1- séquencement des opérations,
E2- langage d' interaction,
E3- dispositifs d' entrée,
E4- dispositifs de présentation,
E5- temps de réponse,
E6- traitement des erreurs,
E7- guidage.

INFORMATICIEN:

Il poursuit un double objectif:

- lors de l' analyse, recherche de spécifications les meilleures possibles,
- lors de la programmation, recherche d' un produit le plus fiable possible.

CORRESPONDANCE des TROIS POINTS de VUE:

Il semble que tous ces paramètres ne se correspondent pas biunivoquement.

Une comparaison UTILISATEUR/ERGONOME met en évidence cette correspondance incomplète:

	E1	E2	E3	E4	E5	E6	E7
U1		*		*			*
U2	*	*	*	*	*	*	
U3	*		*			*	*
U4	*	*	*	*		*	
U5	*						

*: indique la correspondance des points de vue

Une comparaison ERGONOME/INFORMATICIEN permet d' intégrer les connaissances ergonomiques à la conception informatique.

Une approche consisterait à traduire chaque paramètre ergonomique selon la logique de conception utilisée en informatique.

2.2.3.Intégration des Différents Points de Vue.

Dans le développement de l' application, il faut tenir compte de ces différents points de vue.

Le modèle comprend trois parties:

- ANALYSTE

Nous parlons d' une **Représentation Conceptuelle.**

A ce stade, nous avons deux étapes:

* décrire la logique générale du système,
* décrire la logique de traitement par rapport à un poste de travail (éléments de psychologie cognitive liés à l' utilisateur).

- UTILISATEUR

Nous parlons d' une **Représentation Externe.**

A ce stade, nous avons deux étapes:

* traduire les éléments interactifs développés par l' analyste,
* définir tous les éléments propres à cette représentation (éléments d' ergonomie).

- PROGRAMMEUR

Nous parlons d' une **Représentation Interne.**

A ce stade, nous n' avons aucune spécification nouvelle concernant l' utilisateur et donc aucun élément d' ergonomie supplémentaire.

Toutes ces représentations interviennent à un moment ou à un autre dans le processus de développement de l' application.

Il nous apparaît intéressant de donner un **ordre de réalisations de ces étapes:**

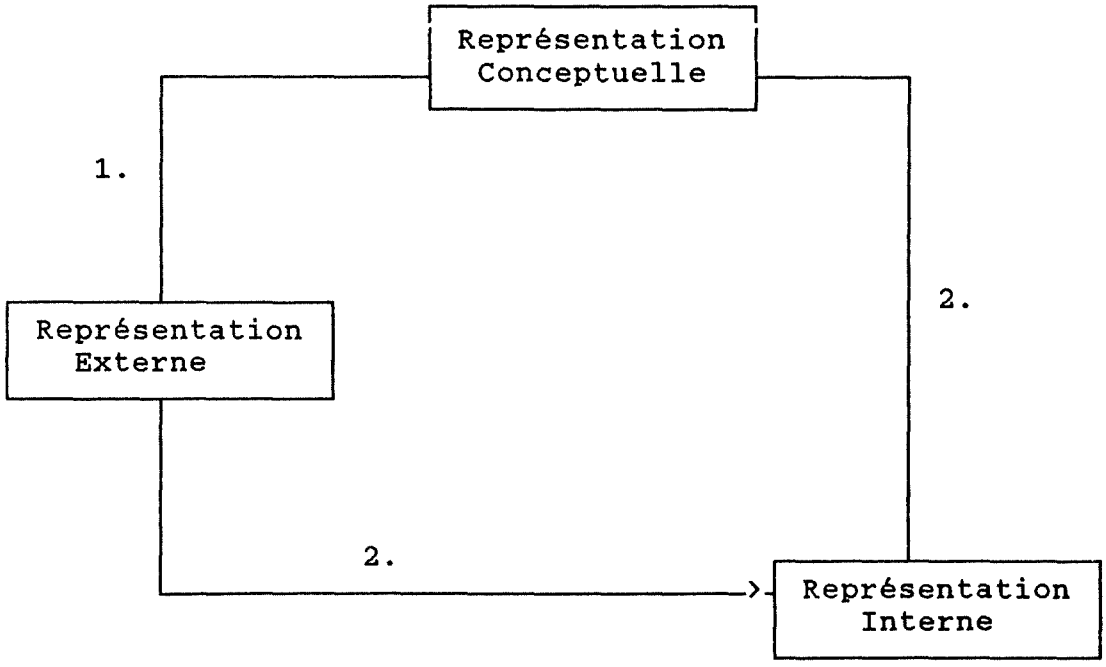


fig. 2.1.

2.2.4.Apport de la psychologie cognitive et de l' ergonomie.

Il est intéressant de connaître les concepts intervenant dans un interface homme-machine.

Liste (non exhaustive) de concepts de **psychologie cognitive:**

P1- Types d' Utilisateurs: tous n' ont pas le même comportement face à un même logiciel,

P2- Logique de Fonctionnement/Utilisation,

P3- Tâches Prévues/Effectives,

P4- Planification Hiérarchique: pensée de la résolution de la tâche par un homme,

Ces quatre concepts ont un impact direct sur la conception de la Représentation Conceptuelle.

P5- Mémoire à Court Terme,

P6- Automatismes: nécessaires pour une partie de l' apprentissage.

Ces deux derniers concepts ont un impact direct sur la conception de la Représentation Externe.

Liste (non exhaustive) de concepts d' ergonomie:

E1- Séquencement des Opérations: permet l' enchaînement des opérations autorisé par le logiciel; cet enchaînement peut être libre ou guidé par l' arborescence d' un menu,

E2- Langage d' Interaction: niveau lexical et syntaxique,

E3- Dispositifs d' Entrée: niveau physique,

E4- Dispositifs de Sortie: niveau affichage,

E5- Temps de Réponse,

E6- Traitement des Erreurs,

E7- Guidage: niveau aide et apprentissage.

Tous ces concepts sont étroitement liés à la Représentation Conceptuelle/Externe:

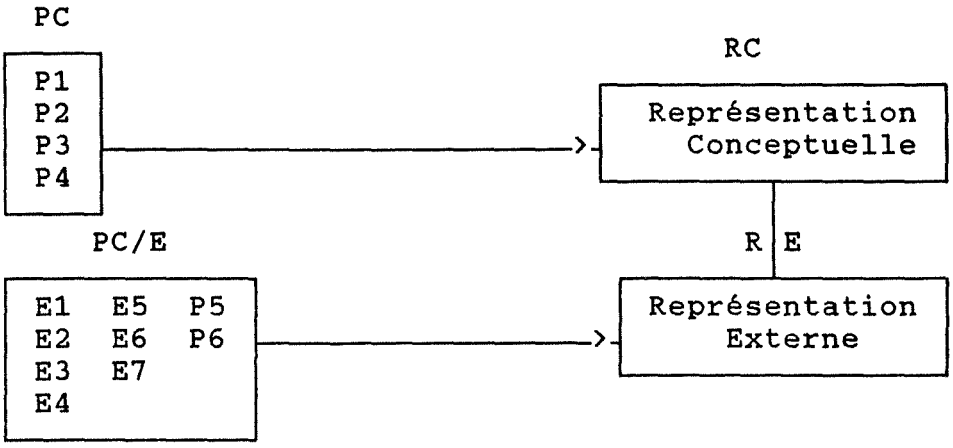


fig. 2.2.

Commentaires:

La conception de la Représentation Externe doit intégrer toutes les caractéristiques de l' utilisateur; certains éléments de psychologie cognitive et d' ergonomie sont intégrés directement et d' autres par le biais de la Représentation Conceptuelle.

Les concepts de psychologie cognitive et d' ergonomie ne sont pas complètement indépendants; ce qui est normal étant donné que leur objet d' étude est le même avec des objets différents.

Ce fait entraîne seulement que des options prises au niveau de la Représentation Conceptuelle ont un impact sur des éléments d' ergonomie, à savoir le séquencement des opérations, le traitement des erreurs et le guidage.

2.3.Structures Profondes des Applications Interactives. [2.2.] [2.6.]

Ce paragraphe reprend les composants PC et RC de la fig. 2.2.

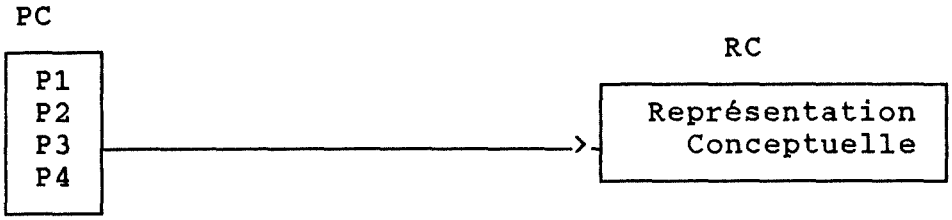


fig. 2.3.

2.3.1.Psychologie Cognitive et Utilisateur de Logiciel.

Il n' est en aucun cas question de faire un exposé sur la psychologie cognitive mais de comprendre le processus mental de l' utilisateur.

Nous allons développer un certain nombre de concepts et plus particulièrement les définir.

1.Définitions.

IMAGE OPERATIVE

Tout utilisateur confronté aux informations portant sur un objet a une représentation mentale de cet objet.

Pour la réalisation d' une tâche, toutes les facettes de cette image ne présentent pas d' intérêt, on en retient une partie et on parle dans ce cas d' une image opérative.

Deux propriétés caractérisent cette image:

- laconisme: l' image étant un moyen d' action, elle ne retient de l' objet que les seules propriétés nécessaires à l' opération,

- déformation fonctionnelle: réplique déformée de l' objet, mais déformée par l' accentuation de ce qui est fonctionnellement important pour une tâche donnée dans un contexte donné.

Il s' agit en tout état de cause d' une approche statique (structure de données).

PLANIFICATION HIERARCHIQUE

Comment l' utilisateur se représente-t-il le travail qu' il a à faire pour atteindre son objectif?

La réponse à cette question permet une utilisation et un apprentissage d' autant meilleurs que le modèle présenté à l' utilisateur se rapproche plus de sa logique d' utilisation.

Il s' agit en tout état de cause d' une approche dynamique (structure de traitements).

Il est une hypothèse faite dans le contexte de la planification hiérarchique estimant que l' utilisateur va élaborer un plan d' action à partir du but qu' il souhaite atteindre.

On définit un plan d' action, c' est-à-dire un certain nombre de sous-buts intermédiaires nécessaires à l' accomplissement du but.

TYPES d' UTILISATEURS

Il existe deux types d' utilisateurs:

- les débutants ou occasionnels,

- les expérimentés.

Différences de Comportement

Il existe une différence au niveau des images opératives.

En effet, plus l' utilisateur est expérimenté, plus il aura une déformation fonctionnelle de l' image opérative.

Au niveau de l' activité cognitive, les expérimentés semblent avoir un jugement plus rapide fondé sur une caractérisation grossière des situations, alors que les débutants cherchent à chiffrer les décisions à prendre de manière très précise.

Apprentissage

Il existe deux logiques:

- logique de fonctionnement:
  décrit les effets de chaque commande
  " si P alors Q " ,

- logique d' utilisation:
  explique à l' utilisateur comment arriver à son but
  " si objectif Q alors P " .

Le passage d' une logique à l' autre est complexe.

Pour réduire le fossé entre l' utilisation et le fonctionnement, on propose de définir des règles d' utilisation qui précisent les actions qu' il est possible de réaliser et comment le faire.

Dans une logique d' utilisation, on pourrait donner l' ensemble des procédures à suivre pour arriver à certains buts mais, dans le cas général, on ne connaît pas à l' avance l' ensemble des tâches pour lesquelles peut être utilisé un dispositif, et donc on ne peut élaborer un ensemble exhaustif de procédures.

Il s' agit donc de définir des actions qui ne correspondent pas à des buts précis mais qui seront des procédures pour la réalisation de ces tâches.

On a ainsi fait pour l' utilisateur le passage des règles de fonctionnement aux composants d' utilisation.

Le problème qui reste, dans le cadre de la logique de fonctionnement, est de déterminer les bonnes règles d' utilisation à faire apprendre, celles qui correspondent à des actions que le sujet se donne comme objectifs intermédiaires.

Pilotage.

Le pilotage d' une application recouvre deux notion:

- contrôle: de l' exécution de la tâche,

- régulation: laisse au pilote la possibilité de modifier certaines conditions d' exécution de la tâche de manière à atteindre l' objectif fixé quels que soient les aléas provenant de l' environnement.

La "latitude décisionnelle" est la part de pilotage qui est sous la responsabilité de l' utilisateur.

Deux extrêmes sont à considérer:

pilotage laissé entièrement à l' utilisateur:

Logiciels bien adaptés pour des postes où les tâches sont peu structurées mais aucun contrôle sur la cohérence de la procédure par rapport au but fixé et l' ordinateur n' est pas utilisé au maximum de ses possibilités (ex: enchaînements automatiques d' écran),

pilotage laissé entièrement à l' ordinateur:

Logiciels garantissant une cohérence maximale du système d' information mais au détriment de la souplesse de mise en oeuvre pour l' utilisateur et de la facilité d' intégrer des variabilités et des aléas.

TACHE et ACTIVITE

De manière générale, la **tâche** indique ce qui est à faire, l' **activité** ce qui se fait.

Plus précisément, la **tâche** est la réalisation d' un but donné dans des conditions déterminées et l' activité est ce qui est mis en oeuvre pour exécuter cette tâche.

La **tâche prévue** est la tâche conçue par celui qui en commande l' exécution.

La **tâche effective** correspond à ce que l' utilisateur fait réellement.

2.Extrapolations de ces Définitions.

Reprenons les concepts cités précédemment et interprétons les dans le contexte de logiciels interactifs.

IMAGE OPERATIVE

- laconisme:

  Important pour la conception des dialogues,

  Il est intéressant d' utiliser un langage de spécialiste-utilisateur dans le cas où les utilisateurs sont des habitués,

  Il n' en est pas de même dans le cas d' utilisateurs novices,

- variation suivant la fonction:

  Par exemple, l' analyste et l' utilisateur risquent d' avoir des visions différentes de l' image opérative d' une tâche à accomplir,

  L' analyste a une vision plus globale et abstraite des procédures de traitement que l' utilisateur,

- variation suivant expérience:

Des présentations doivent être différentes pour les utilisateurs novices et les utilisateurs confirmés,

PLANIFICATION HIERARCHIQUE

Description orientée-but qui part du résultat que veut atteindre l' utilisateur pour remonter jusque aux actions élémentaires nécessaires à la réalisation du but.

LOGIQUE FONCTIONNEMENT / UTILISATEURS

Il est important de constater que la plupart des logiciels interactifs sont élaborés et présentés à l' utilisateur selon une logique de fonctionnement.

Ceci proviendrait du fait que les méthodes courantes d' analyse fournissent des descriptions fonctionnelles plutôt que des logiques d' utilisation.

TACHE PREVUE / EFFECTIVE

On remarque que si, lors des interviews, on a essentiellement interrogé des responsables, on a beaucoup de chances d' avoir recueilli des tâches prévues; par contre, si on a interrogé des utilisateurs, on a plus de chances d' avoir recueilli des tâches effectives ou un mélange de tâches effectives et de tâches prévues.

3.Interaction entre ces Différents Concepts.

Les concepts pré-cités ne sont pas indépendants.

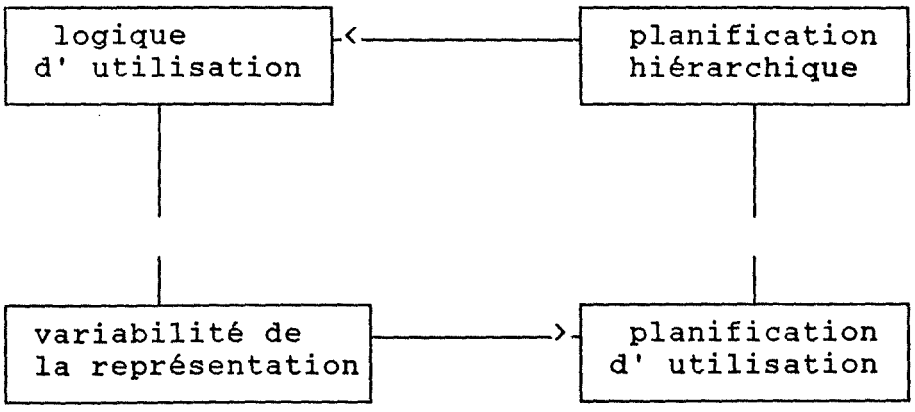


fig. 2.4.

Commentaires:

La planification hiérarchique permet de trouver la logique d' utilisation pour un but donné.

Les variabilités de la représentation se traduisent soit par des logiques d' utilisation différentes, soit par les notions de tâches prévues et de tâches effectives.

Pour décrire les tâches prévues et effectives, on peut utiliser la planification hiérarchique.

2.3.2.Aides à l' Utilisateur.

Définissons des aides à l' utilisateur qui ont un caractère commun à toutes les applications interactives.

Il existe en effet des paramètres constants (valables pour toutes les applications) et variables (propres à une application).

1.Aides au Travail.

INTERRUPTION

L' utilisateur peut quitter n' importe quelle opération en cours d' exécution pour aller exécuter une autre opération et reprendre la première opération à son point d' interruption.

TRANSFERT des DONNEES

L' utilisateur peut transférer une donnée d' une opération à une autre sans avoir à la ressaisir.

On minimise ainsi les saisies et donc les erreurs de recopie.

QUITTER

L' utilisateur doit pouvoir demander l' abandon de son travail à n' importe quel moment sans avoir à finir une opération ou à repasser par une arborescence du menu.

Le logiciel peut lui signaler les conséquences de cette demande selon l' endroit où elle se situe.

DIFFERER

Semblable à QUITTER mais avec un processus de mémorisation de l' opération non terminée avec possibilité de la reprendre.

ANNULER

Annulation de la dernière action.

MEMORISATION de l' ACTIVITE

L' utilisateur peut avoir des difficultés pour mémoriser ce qu' il a entré.

2.Aide à l' Apprentissage du Logiciel.

GUIDAGE FONCTIONNEL

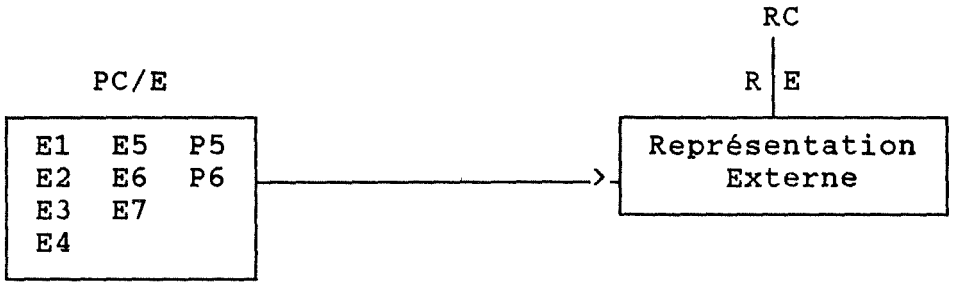
Une commande de type S.O.S. doit permettre à tout moment à l' utilisateur de connaître soit la liste des opérations possibles dans l' état actuel d' avancement de son travail, soit une explication des fonctions et effets d' une commande donnée.

GUIDAGE d' UTILISATION

Guidage devant répondre à la question "comment faire pour ?" .

2.4.Aspects de Surface des Applications Interactives. [2.2.] [2.6.]

Ce paragraphe reprend les composants PC/E et RE de la fig. 2.2.



On présente ce que voit l' utilisateur de l' application interactive.

2.4.1. Apport de la Psychologie Cognitive à la RE.

1.Mémoires Humaines.

On distingue trois mémoires:

- registre de l' information sensorielle:

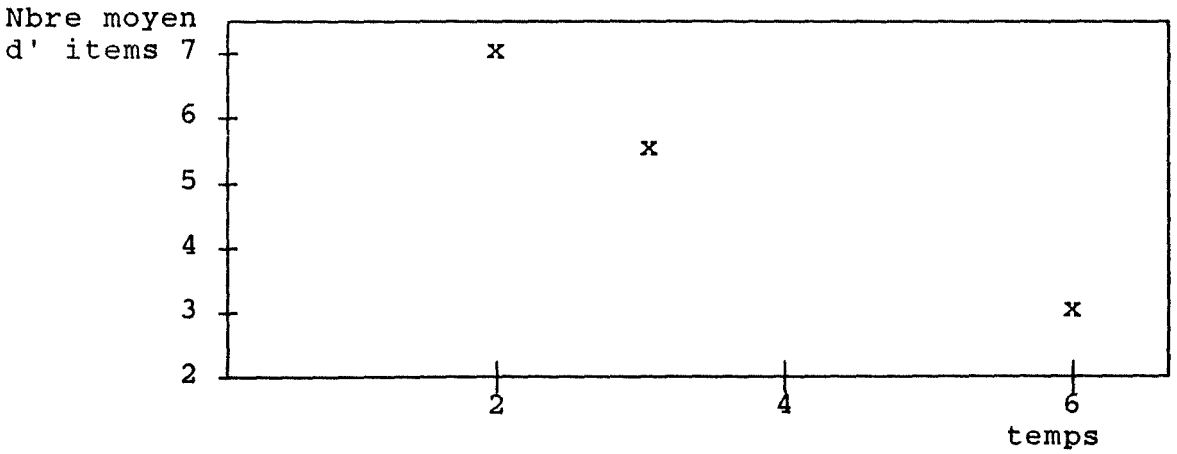
  Capable de contenir l' intégralité d' une image pendant un temps très court (0.1 s. à 0.5 s.),(1)

- mémoire à court terme:

  Extrait les éléments significatifs des registres de l' information sensorielle,

  Limite de l' être humain: 7 items en 2s.

  Diagramme d' effacement de la mémoire à court terme sans aucune influence externe:



---

(1) [2.5] Dans le système international (S.I.), l' abrévi-ation de la "seconde" est "s.".

En cas d' autorépétition, on peut conserver l' information en mémoire à court terme beaucoup plus longtemps,

Si on est victime d' interférence, l' information quitte plus rapidement la mémoire à court terme et est perdue à jamais si on n' a pas eu le temps de faire basculer cette information en mémoire à long terme,

- mémoire à long terme:

Sans limite du point de vue durée de vie,

L' utilisation de la mémoire à long terme requiert de la concentration qui rend l' individu indisponible pour d' autres tâches,

D' autre part, la mémorisation à long terme est facilitée par la compréhension et par une logique de rattachement à quelque chose de connu et donc les analogies.

Une information structurée est plus facilement mémorisable.

Conséquences:

- il faut remédier aux effacements de la mémoire à court terme dus à des situations imprévues (ex: téléphone) en mémorisant par exemple la dernière action de l' utilisateur,

- réduire le temps de réponse de l' ordinateur pour réduire les effets de la dégradation de la mémoire à court terme,

- rendre les informations à retenir les plus concises possibles pour réduire les effets de la limite de capacité de la mémoire à court terme,

- travailler par analogies pour améliorer les capacités de la mémoire à long terme.

2.Automatismes.

L' utilisateur habitué à effectuer les mêmes tâches sur des périodes suffisamment longues acquiert des automatismes qui lui permettent d' exécuter ces tâches très rapidement et avec une mobilisation minimale de ces processus conscients.

On cherche à créer des interfaces suscitant de plus en plus les automatismes.(1)

---

(1)Exemple:
la touche validant les entrées des données doit être la même pour toute les applications informatiques.

2.4.2.Apport de l' Ergonomie à la RE.

1.Introduction.

Comme vu au chapitre 2.2.2., l' ergonome décompose son problème en fonction de sept paramètres.

Mais ces paramètres ne sont pas homogènes, et en particulier, ils ont des impacts différents sur les utilisateurs.

En effet, un utilisateur peut être décrit en trois niveaux:

I- caractéristiques physiologiques,

II- traitement par l' utilisateur des opérations élémentaires,

III- système de représentation et de traitement de l' information de l' utilisateur qui lui permet de synthétiser, résoudre un problème en fonction des informations disponibles.

Nous pouvons représenter l' ensembles des effets:

niveaux / param.	I	II	III
E1		x	x
E2		x	x
E3	x		
E4	x	x	
E5			x
E6		x	x

Nous nous intéressons uniquement aux niveaux I et II qui relèvent de l' ergonomie du logiciel.

2.Recommandations Ergonomiques pour les Paramètres.

Il s' agit de RECOMMANDATIONS et non pas de règles.

Une bonne expérience est préférable à ces recommandations.

De plus, la liste des recommandations est non exhaustive.

E1-SEQUENCEMENT des OPERATIONS

Le problème évoqué est l' adéquation entre l' ordre des opérations fixé par la machine et celui qui est nécessaire à l' opérateur pour effectuer sa tâche dans n' importe quel environnement.

En effet, on a tendance à toujours résoudre un problème suivant le déroulement normal (tâche prévue) sans tenir compte du type d' utilisateur ou d' aléas provenant de l' environnement; alors qu' il s' agit du contexte naturel de toute tâche effectuée par un être humain.

Les caractéristiques du séquencement font partie de la structure profonde du logiciel.

Il s' agit de:

- type d' enchaînement (libre, guidé, automatique),
- possibilités d' enchaînements variés selon le degré d' expérience et le type d' utilisateur,
- accéder à toutes les informations avec autant ou plus de facilité qu' à la main,
- pouvoir quitter, annuler, interrompre à tout moment,
- afficher ce qu' on vient de faire afin de reprendre plus facilement après interruption,
- faire appel à une opération quelconque à partir d' une autre et de revenir en arrière.

E2-LANGAGE d' INTERACTION

Outil permettant à l' utilisateur d' exprimer, au travers d' un vocabulaire et d' une syntaxe, les opérations qu' il désire effectuer.

VOCABULAIRE

Il est préférable d' utiliser le langage du spécialiste de la tâche plutôt que celui de l' informaticien.

Dans le cas où on ne peut pas utiliser des mots du vocabulaire courant ou professionnel en leur conservant le même sens, il existe d' autres vocabulaires de langages de commande:

- Codes Numériques

avantages:

* courts,

* rapides à saisir,

désavantages:

* difficulté de mémorisation,

* risque d' erreurs élevés,

- Codes Mnémoniques

Laisser aux utilisateurs le soin de créer leurs abréviations,

- Mots en Langue Française

Paraît être une bonne solution,

L' évolution d' un langage [2.3.] développé dans le cadre d' une messagerie électronique est intéressant car le résultat est absolument contre-intuitif,

Voici les principales conclusions:

- le vocabulaire est mieux compris par les informaticiens que par les utilisateurs naïfs; en effet ce vocabulaire a été choisi par l' informaticien et les mots utilisés peuvent avoir un sens différent dans le contexte informatique,

- les termes choisis nécessitent un apprentissage,

- il y a des confusions dues aux synonymes,

- il est plus difficile de mémoriser une nouvelle définition d' un mot connu que de mémoriser un nouveau mot,

- Mots Inconnus (créé ou issu d' une langue étrangère)

avantage:

* peu de risque de confusion,

désavantage:

* plus gros effort de mémorisation,

- Menu glissant qui permet de visualiser une liste de commandes plus grande que la place disponible,

- Menu dynamique qui signale à l' utilisateur les commandes disponibles à un moment donné de l' interaction (au moyen de surbrillance par exemple).

Cet ensemble de présentations demande peu de mémorisation de la part des utilisateurs; il s' agit en effet de menus affichés de manière permanente.

Dans le cas d' affichage permanent, la ligne où est affichée le menu a un contenu évolutif selon la demande de l' utilisateur; au premier niveau, elle contient la liste des buts et au dernier niveau, la liste des opérations.

Dans le cas du multifenêtrage, on peut avoir une variante qui consiste à mettre la liste des opérations dans un menu déroulant.

L' apparition et le développement du multifenêtrage s' explique particulièrement pour visualiser simultanément plusieurs opérations.

Pour sélectionner les commandes, on peut utiliser diverses possibilités:

- Positionner le curseur sur la commande au moyen des touches-fonctions de déplacement (ou d' une souris, ...) puis valider par une touche-fonction,
  Cette possibilité demande peu d' efforts de mémorisation et est donc adaptée aux novices,

- Taper la première lettre de la commande en association avec une autre touche indiquant qu' on est en mode saisie de commande,
  Cette possibilité demande plus d' efforts de mémorisation et s' applique plutôt aux habitués,

- Utiliser des touches-fonctions associées à chaque commande,
  Cette possibilité est très efficace pour des utilisateurs expérimentés,

E5-TEMPS de REPONSE

Il s' agit de déterminer le temps de réponse idéal.

Etant donné la configuration de la mémoire humaine, il est gênant d' être interrompu pendant la période de mémorisation ainsi que de devoir conserver l' information dans la mémoire à court terme plus de 4s.

Par contre, un temps de réponse plus long pour une opération ne demandant pas de mémorisation ne pose pas de problème à condition que l' utilisateur soit averti par un message.

Si le temps de réponse est vraiment trop long, il faut afficher ce temps pour que l' utilisateur puisse faire éventuellement autre chose en attendant.

Conclusion:

- moins de 2s.: temps de réponse idéal,

- 2s. à 4s.: impression d' attente, peu gênante pour la mémoire à court terme,

- plus de 4s.: trop long si le dialogue nécessite une mémorisation à court terme et s' il n' y a pas de message affiché à l' écran.

E6-TRAITEMENT des ERREURS

Il existe deux types d' erreurs:

- Erreurs d' Exécution: correspond aux erreurs de frappe qui sont faciles à corriger,
  Dans ce type d' erreur, l' utilisateur peut appeler soit une commande inexistante soit une commande indisponible à ce moment,
  Pour réduire les erreurs de ce derniers type, il est utile d' indiquer celles qui sont activables ou non,

- Erreurs d' Intention: correspond aux erreurs de mauvaise compréhension des commandes qui sont plus difficiles à détecter et dont la correction demandent un apprentissage de la part de l' utilisateur,

On note un certain nombre de considérations sur le signalement des erreurs:

- signalement le plus rapide possible à cause de la volatilité de la mémoire à court terme,

- les messages d' erreur les plus explicatifs,

- les messages d' erreur se présentent sous la même forme.

En ce qui concerne l' emplacement du texte de l' erreur, il existe plusieurs possibilités:

- à côté de l' erreur s' il y a assez de place ou si on travaille en multifenêtrage ,

- dans une zone réservée, mais en remarquant que l' utilisateur risque de ne pas le voir si on n' ajoute aucune marque spéciale (sonnerie, ...).

Pour les corrections d' erreurs, l' utilisateur doit pouvoir revenir aisément à l' opération ou la ligne où se situe l' erreur.

On retombe sur des problèmes sur la structure profonde du logiciel.

E7-GUIDAGE

On distingue deux types de guidages:

- fonctionnel par description des commandes: décrit les effets d' une commande, ses conditions d' application; convient aux utilisateurs expérimentés qui veulent se rappeler la syntaxe et la sémantique d' une commande.

- d' utilisation par reconnaissance des plans d' action de l' utilisateur: convient aux utilisateurs novices ou intermittents car il peut les guider dans la réalisation d' une tâche en leur indiquant l' ensemble des opérations à parcourir en fonction du but qu' ils veulent atteindre.

2.5.Structure Interne des Logiciels Interactifs. [2.6.]

Cette partie est consacrée à la traduction de la RC et la RE.

Elle n' apporte rien de nouveau par rapport à la RC et la RE.

Nous allons présenter les fonctionnalités d' un système de gestion multifenêtre (SGM) du point de vue de l' utilisateur puis du point de vue du programmeur.

2.5.1.Fonctionnalités d' un SGM-Utilisateur.

Les fonctions principales d' un SGM sont de permettre l' accès en parallèle à différentes applications et la communication entre applications.

La communication entre applications présente deux aspects:

- démarrage et arrêt des applications,

- échange d' informations entre applications actives.

1.Conséquences de ces deux Fonctionnalités.

GESTION des MENUS

Le menu peut apparaître constamment à l' écran ou uniquement à la demande de l' utilisateur.

Le menu peut être modifié interactivement par l' activité et ainsi les commandes permises à ce moment seraient signalées de façon particulière.

GESTION des DESIGNATIONS

La désignation permet à l' utilisateur d' identifier un point sur l' écran.

La sélection permet à l' utilisateur d' identifier une entité logique.

Une sélection peut être définie par une ou plusieurs désignations.

Les sélections peuvent se faire de deux façons:

- désigner l' information de façon spatiale,

- utiliser la structure hiérarchique de cette information.

GESTION de l' ECHO

Rassure l' utilisateur sur son action.

Plusieurs types de réponses existent:

- résultat d' une opération,
- surbrillance de l' objet choisi,
- signaler le travail de l' ordinateur,
- signaler l' attente de paramètres "utilisateur".

GESTION des ERREURS

Le système gère deux fonctions:

- signalement des erreurs dans une fenêtre ou près d' une zone erronée,
- commandes de rectification d' erreurs telles que:
  * " annuler " pour une exécution en cours,
  * " arrêter " pour arrêt temporaire d' exécution,
  * " reprendre " après un arrêt temporaire,
  * " défaire " pour se trouver dans l' état précédent d' activation d' une commande.

GESTION du SIGNAL

Elle permet d' indiquer à l' utilisateur qu' un événement vient de parvenir à l' ordinateur.

AIDE à l' UTILISATEUR

A tout moment, l' utilisateur peut avoir besoin d' explications sur une commande particulière:

- ensemble des fonctions accessibles à un moment donné,
- documentation complète du système,
- causes possibles d' erreurs.

2.Mise en Oeuvre de ces Fonctions.

DEMARRAGE et ARRET du SYSTEME

On peut identifier au démarrage l' utilisateur; dans ce cas, on peut connaître les dernières actions de l' utilisateur.

GESTION des FENETRES

La mise en oeuvre des fenêtres la plus simple pour l' utilisateur consiste à créer automatiquement une fenêtre dès qu' il active une nouvelle activité et inversement à détruire automatiquement cette fenêtre lorsque cette activité est considérée comme terminée.

Une fenêtre comprend deux parties:

- clôture: partie de la fenêtre délimitant l' espace où on peut affecter l' information de l' activité; si cette information est plus grande que l' espace délimité par la clôture, il faut prévoir un système de défilement commandé par l' utilisateur,

- cadre: permet de distinguer les fenêtres les unes des autres.

La gestion des fenêtres implique le partage de l' écran qui a deux limites:

- taille et position de la fenêtre choisies par l' utilisateur,

- taille et position de la fenêtre choisies par le système qui peut optimiser l' espace de travail.

Plus on crée de fenêtres, plus la taille de chaque fenêtre diminue.

On peut se situer entre les deux cas extrêmes.

Il faut être attentif aux liens hiérarchiques entre fenêtres:

- lien mère-fille: une opération (fermeture, ouverture, déplacement) sur la fenêtre mère entraîne automatiquement la même opération sur la fenêtre fille,

- lien frère-soeur: l' utilisateur ne peut travailler que sur une des soeurs à la fenêtre.

SYNTAXE de COMMANDES

- Commandes préfixées:

1. sélectionner la commande,
2. sélectionner l' objet à traiter.

Syntaxe considérée comme la plus naturelle.

Après chaque commande, l' utilisateur ne peut spécifier que les objets portant sur cette commande.

Cette syntaxe présente donc un inconvénient si l' utilisateur veut utiliser plusieurs commandes et si on veut conserver la simplicité de la syntaxe.

- Commandes postfixées:

1: sélectionner l' objet à traiter,
2: sélectionner la commande.

Syntaxe ne s' adaptant pas aux commandes à plusieurs paramètres.

- Commandes infixées:

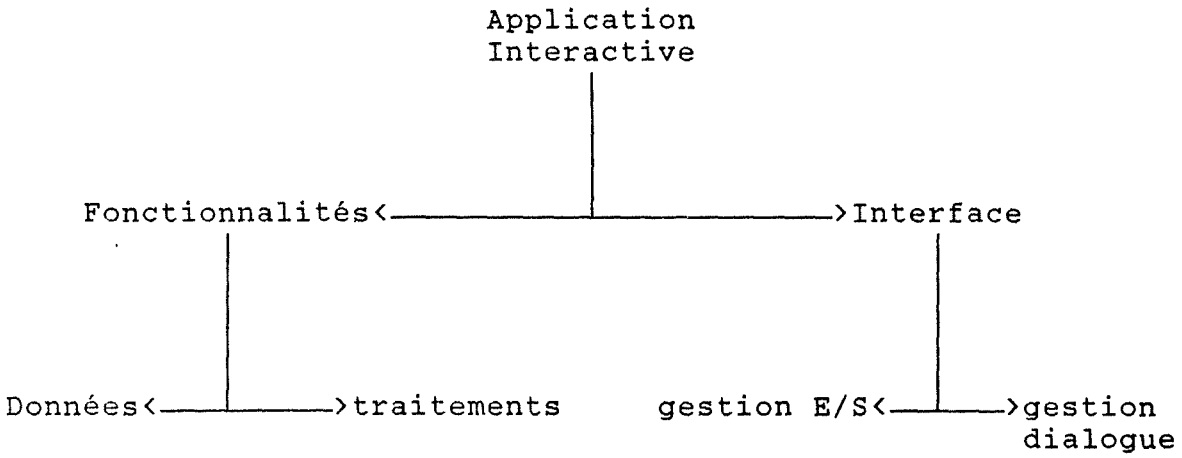
1: sélectionner l' objet à traiter,
2: sélectionner la commande,
3: sélectionner l' objet à traiter.

Syntaxe intermédiaire entre les deux précédentes.

Les paramètres entourent la commande, ce qui rend simple une commande à deux paramètres, sinon on cumule les inconvénients des deux autres syntaxes.

2.5.2.Fonctionnalités d' un SGM-Programmeur.

Il est important de séparer au niveau du code les fonctionnalités concernant l' application de celles concernant l' interface homme-machine.



Les fonctionnalités sont décomposées en données et traitements, d' une part pour la facilité des modifications ultérieures, d' autre part pour la minimisation de la programmation en utilisant des systèmes de gestion de base de données.

L' interface est subdivisé en gestion d' E/S, totalement indépendante de l' application, et gestion du dialogue qui fait le lien entre les E/S et les fonctionnalités de l' application.

2.6.Bibliographie.

[2.1.] J.P.E. CASSAR et L.G. GARCEAU et Th. BARIBEAU

La BUREAUTIQUE: Planification Implantation Gestion
Le guide de l' entreprise

Ed. Organisation - 1988

[2.2.] M.-F. BARTHET

LOGICIELS INTERACTIFS et ERGONOMIQUES
Modèles et Méthodes de Conception

Dunod Informatique - 1988

[2.3.] D. SCAPIN

Guide Ergonomique de Conception des Interfaces
Homme-Machine

Rapport INRIA n° 77 - 1987

[2.4.] SPERANDIO - BOUJU

L' Exploration Visuelle de Données Numériques
Présentées Sur Ecran Cathodique

Le Travail Humain - Tome 46, n° 1 - 1983

[2.5.] NBN X02 - 001 - 2e édition, Décembre 1974

Unités et Symboles

[2.6.] B. SHNEIDERMAN

DESIGNING the USER INTERFACE
Strategies for Effective Human-Computer Interaction

Addison-Wesley Publishing Company - 1987

DEVELOPPEMENT de l' APPLICATION

3.1.Définition du Problème.

Décrivons le fonctionnement d' un établissement scolaire du niveau secondaire.

Un établissement scolaire est constitué d' un ensemble de N **étudiants**; cet ensemble forme 1 et 1 seule **section.**
Un étudiant suit 1-N **cours à option.**
Une section suit 1-N **cours du tronc commun.**
Un **professeur** donne 0-1-N cours à option et/ou 0-1-N cours du tronc commun.
Un cours se passe dans 1-N **local(locaux).**
Un cours est donné à 1-N **heure(s)** de la semaine.

Décrivons les différents éléments intervenant dans un établissement scolaire du niveau secondaire.

Un(e) **étudiant(e)** a un numéro identifiant,un nom, un prénom.
Une **section** a un numéro identifiant, un nombre d' étudiants la constituant.
Un **cours** a un numéro identifiant, un nom, une contrainte local-activité
- exemple1: le cours de physique doit se donner dans un local de physique
- exemple2: le cours de gymnastique doit se donner dans un gymnase

Un **local** a un numéro identifiant, une contrainte activité-local, une contrainte d' indisponibilité
- exemple1: le gymnase ne peut recevoir que le cours de gymnastique
- contre-exemple1: le laboratoire de physique peut accueillir d' autres cours que le cours de physique

Une **heure** est identifiée par un numéro dans la semaine.
Un **professeur** a un numéro identifiant, un nom, un prénom, une contrainte d' indisponibilité.

Décrivons les contraintes implicites à respecter.

Un professeur ne peut donner plusieurs cours à une même heure.
Un professeur ne peut se trouver dans plusieurs locaux à une même heure.
Un professeur peut donner un cours du tronc commun à plusieurs classes dans un même local à une même heure.
Un professeur peut donner un cours à option à plusieurs étudiant(e)s dans un même local à une même heure.
Un(e) étudiant(e) appartient à une et une seule section.
Un(e) étudiant(e) ne peut suivre plusieurs cours à option à la même heure.
Une section ne peut suivre plusieurs cours du tronc commun à la même heure.
Une section et un(e) étudiant(e) appartenant à cette section ne peuvent suivre respectivement un cours du tronc commun et un cours à option à la même heure.

3.2.Schéma Conceptuel.

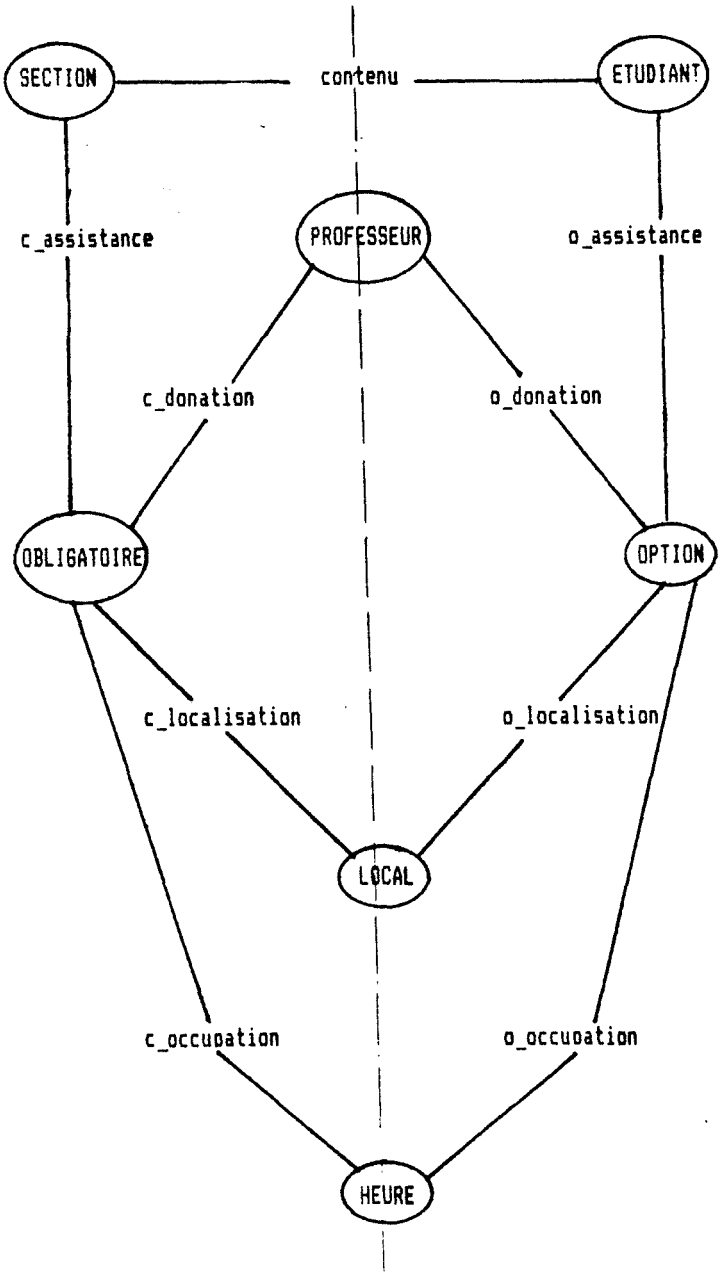
3.2.1.Schéma Entité-Association.

Le paragraphe précédent va nous permettre de concevoir la modélisation du problème.

Celle-ci se fera selon le modèle ENTITE-ASSOCIATION.

Nous nous contenterons, pour des raisons de clarté, de représenter les TE(1) et les TA(2).

Les contraintes implicites s' exprimeront in extenso.



(1) TE = Type d' Entité
(2) TA = Type d' Association

3.2.2.Schéma Conceptuel.

Entité: SECTION

Définition:
Ensemble d' étudiants regroupés selon des règles propres à l' établissement scolaire considéré.

Identifiant:
numsec: numéro de la section.

Attribut:
taillesec: taille de la section, c'-est-à-dire nombre d' étudiants constituant la section.

Entité: ETUDIANT

Définition:
Personne répondant aux exigences administratives d' un établissement scolaire et souhaitant s' y inscrire.

Identifiant:
numet: numéro de l' étudiant dans l' établissement.

Attributs:
nomet: nom de famille de l' étudiant.
prenet:prénom de l' étudiant.

Entité: OBLIGATOIRE

Définition:
Cours suivis par l' ensemble des étudiants d' une ou plusieurs sections suivant la population de celle-ci.

Identifiant:
numob: numéro du cours du tronc commun.

Attributs:
nomob: nom du cours du tronc commun.
ex_cont_loc_ob: existence d' une contrainte local-activité.
cont_loc_ob: locaux associés à cette activité.
heure_ob: nombre d' heures pour ce cours du tronc commun.

Entité: OPTION

Définition:
Cours suivis par un ou plusieurs étudiants appartenant à une ou plusieurs sections sans pour autant que celles-ci soit complètement représentées.

Identifiant:
numop: numéro du cours à option.

Attributs:
nomop: nom du cours à option.
ex_cont_loc_op: existence d' une contrainte local-activité. cont_loc: locaux associés à cette activité.
heure_op: nombre d' heures pour ce cours à option.

Entité: PROFESSEUR

Définition:
Personne engagée par le pouvoir organisateur (en ce compris l'Etat) pour dispenser des cours (à option et/ou du tronc commun).

Identifiant:
numpr: numéro du professeur.

Attributs:
nompr: nom de famille du professeur.
prenpr: prénom du professeur.
ex_cont_ind_pr: existence d' une contrainte d' indisponibilité du professeur.
heure_ind_pr: heure d' indisponibilité du professeur.

Entité: LOCAL

Définition:
Pièces de l' établissement dans laquelle se déroulent les cours.

Identifiant:
num_loc: numéro du local

Attributs:
ex_cont_act: existence d' une contrainte activité-local.
cont_act: numéro des activités associées à ce local.
ind_loc: heure d' indisponibilité du local.
taille_loc:nombre de places aménagées dans le local.

Entité: HEURE

Définition:
Unité de temps dans le cadre d' un horaire.

Identifiant:
numheure: numéro de l' heure dans la semaine.

Association: CONTENU

Définition:
Associe à 1 section 1-N étudiant(e)s.

Association: C_ASSISTANCE

Définition:
Associe à 1 section 1-N cours du tronc commun.

Association: O_ASSISTANCE

Définition:
Associe à 1 étudiant(e) 1-N cours à option.

Association: C_DONATION

Définition:
Associe à 1 cours du tronc commun 1-N professeurs.

Association: O_DONATION

Définition:
Associe à 1 cours à option 1-N professeurs.

Association: C_LOCALISATION

Définition:
Associe à 1 local 1-N cours du tronc commun.

Association: O_LOCALISATION

Définition:
Associe à 1 local 1-N cours à option.

Association: C_OCCUPATION

Définition:
Associe à 1 heure 1-N cours du tronc commun.

Association: O_OCCUPATION

Définition:
Associe à 1 heure 1-N cours à option.

3.2.3.Découpe en Fonctions.

Nous relèverons dans le cadre de ce travail 6 fonctions décrivant des actions élémentaires pour l' utilisateur.

Ces 6 fonctions se décomposent en 2 groupes:

1.Fonctions de Mise à Jour de la B.D.

Fonction: FORMATION

Objectifs:
Cette fonction permettra à l' utilisateur de créer des chemins entre les différents TE dont les valeurs ont été préalablement mises à jour.

Messages en entrée:
numsec: numéro de la section.
numet: numéro de l' étudiant.
numob: numéro du cours du tronc commun.
numop: numéro du cours à option.
numpr: numéro du professeur.
num_loc: numéro du local.
numheure: numéro de l' heure.

Contraintes:
Les données vérifient les contraintes suivantes:
CT1: il ne peut y avoir que 1 et 1 seul "numob" ou"numop" pour 1 "numpr + numheure";
CT2: il ne peut y avoir que 1 et 1 seul "num_loc" pour 1 "numpr + numheure";
CT3: il ne peut y avoir que 1 et 1 seul "numsec" pour 1 "numet";
CT4: il ne peut y avoir que 1 et 1 seul "numop" pour 1 "numet + numheure";
CT5: il ne peut y avoir que 1 et 1 seul "numob" pour 1 "numsec + numheure";
CT6: il ne peut pas y avoir "numsec + numet + numheure" pour "numop + numob".

Messages de sortie:
sortie1: "la taille de la section est supérieure à la taille du local";
sortie2: "local mal adapté aux activités";
sortie3: "professeur indisponible";
sortie4: "activité mal adaptée au local";
sortie5: "local indisponible";
sortie6: "professeur occupé";
sortie7: "section occupée";
sortie8: "étudiant(e) occupé(e)";
sortie9: "local occupé";
sortie10:"saturation".

Effets sur le S.I.:
Les changements possibles du S.I. dus à cette fonction sont:

E1: création de chemins entre les différents TE dont les valeurs ont été préalablement mises à jour.
Cette création se réalise uniquement si elle porte sur un chemin complet du schéma E/A.

Règles de traitements:

R1: si taillesec > taille_loc alors sortie1;
R2: si ex_cont_loc_ob ou ex_cont_loc_op est positive; si num_loc n' appartient pas à cont_loc_ob ou cont_loc_op alors sortie2;
R3: si ex_cont_ind_pr est positive; si numheure associée au professeur appartient à heure_ind_pr alors sortie3;
R4: si ex_cont_act est positive; si numop ou numob n' appartient pas à cont_act alors sortie4;
R5: si numheure associée à un local appartient à ind_loc alors sortie5;
R6: si à une heure donnée, on associe un professeur et un ou plusieurs cours à option ou obligatoires alors sortie6;
R7: si à une heure donnée, on associe une section et un ou plusieurs cours obligatoires alors sortie7;
R8: si à une heure donnée, on associe un(e) étudiant(e) et un ou plusieurs cours à option alors sortie8;
R9: si à une heure donnée, on associe un local à un ou plusieurs cours à option ou obligatoires alors sortie9;
R10: si un élément de l' horaire est entièrement défini, la création d' un chemin provoquera la sortie10;
R11: si les règles R1, R7, R8, R9 n' ont pas été appliquées alors E1;

Fonction: SUPPRESSION

Objectifs:

Cette fonction permettra à l' utilisateur de supprimer des chemins entre les différents TE dont les valeurs ont été préalablement mises à jour.
Ce traitement interviendrait lors d' une erreur de manipulation.

Messages en entrée:

numsec: numéro de la section.
numet: numéro de l' étudiant.
numob: numéro du cours du tronc commun.
numop: numéro du cours à option.
numpr: numéro du professeur.
num_loc: numéro du local.
numheure: numéro de l' heure.

Contraintes:

Les données vérifient les contraintes suivantes:

CT1: il ne peut y avoir que 1 et 1 seul "numob" ou"numop" pour 1 "numpr + numheure";
CT2: il ne peut y avoir que 1 et 1 seul "num_loc" pour 1 "numpr + numheure";
CT3: il ne peut y avoir que 1 et 1 seul "numsec" pour 1 "numet";
CT4: il ne peut y avoir que 1 et 1 seul "numop" pour 1 "numet + numheure";
CT5: il ne peut y avoir que 1 et 1 seul "numob" pour 1 "numsec + numheure";
CT6: il ne peut pas y avoir "numsec + numet + numheure" pour "numop + numob".

Effets sur le S.I.:

Les changements possibles du S.I. dus à cette fonction sont:

E1: suppression de chemins entre les différents TE de l' horaire.
Cette suppression se réalise uniquement si elle porte sur un chemin complet du schéma E/A.

Règles de traitements:

R1: E1;

Fonction: MODIFICATION

Objectifs:

Cette fonction permettra à l' utilisateur de modifier (souvent quand il y a "blocage") les chemins entre les différents TE dont les valeurs ont été préalablement mises à jour.
Il faut pouvoir revenir en arrière et défaire ce qui a été fait.

Messages en entrée:

numsec: numéro de la section.
numet: numéro de l' étudiant.
numob: numéro du cours du tronc commun.
numop: numéro du cours à option.
numpr: numéro du professeur.
num_loc: numéro du local.
numheure: numéro de l' heure.

Contraintes:
Les données vérifient les contraintes suivantes:

CT1: il ne peut y avoir que 1 et 1 seul "numob" ou"numop" pour 1 "numpr + numheure";
CT2: il ne peut y avoir que 1 et 1 seul "num_loc" pour 1 "numpr + numheure";
CT3: il ne peut y avoir que 1 et 1 seul "numsec" pour 1 "numet";
CT4: il ne peut y avoir que 1 et 1 seul "numop" pour 1 "numet + numheure";
CT5: il ne peut y avoir que 1 et 1 seule "numob" pour 1 "numsec + numheure";
CT6: il ne peut pas y avoir "numsec + numet + numheure" pour "numop + numob".

Messages de sortie:

sortie1: "la taille de la section est supérieure à la taille du local";
sortie2: "local mal adapté aux activités";
sortie3: "professeur indisponible";
sortie4: "activité mal adapté au local";
sortie5: "local indisponible";
sortie6: "professeur occupé";
sortie7: "section occupée";
sortie8: "étudiant(e) occupé(e)";
sortie9: "local occupé";
sortie10:"saturation".

Effets sur le S.I.:
Les changements possibles du S.I. dus à cette fonction sont:

E1: création de chemins entre les différents TE.
E2: suppression de chemins entre les différents TE.

Règles de traitements:

R1: si taillesec > taille_loc alors sortie1;

R2: si ex_cont_loc_ob ou ex_cont_loc_op est positive; si num_loc n' appartient pas à cont_loc_ob ou cont_loc_op alors sortie2;

R3: si ex_cont_ind_pr est positive; si numheure associée au professeur appartient à heure_ind_pr alors sortie3;

R4: si ex_cont_act est positive; si numop ou numob n' appartient pas à cont_act alors sortie4;

R5: si numheure associée à un local appartient à ind_loc alors sortie5;

R6: si à une heure donnée, on associe un professeur et un ou plusieurs cours à option ou obligatoires alors sortie6;

R7: si à une heure donnée, on associe une section et un ou plusieurs cours obligatoires alors sortie7;

R8: si à une heure donnée, on associe un(e) étudiant(e) et un ou plusieurs cours à option alors sortie8;

R9: si à une heure donnée, on associe un local à un ou plusieurs cours à option ou obligatoires alors sortie9;

R10: si un élément de l' horaire est entièrement défini, la création d' un chemin provoquera la sortie10;

R11: si les règles R1, R7, R8, R9 n' ont pas été appliquées alors E1 puis E2;

2.Fonctions de Consultation de la B.D.

Fonction: EDITER

Objectifs:
Cette fonction permettra à l' utilisateur de prendre connaissance du contenu des différents TE.

Messages en entrée:
numsec: numéro de la section.
numet: numéro de l' étudiant.
numob: numéro du cours du tronc commun.
numop: numéro du cours à option.
numpr: numéro du professeur.
num_loc: numéro du local.
numheure: numéro de l' heure.

Contraintes:
Les données vérifient les contraintes suivantes:
CT1: il ne peut y avoir que 1 et 1 seul "numob" ou"numop" pour 1 "numpr + numheure";
CT2: il ne peut y avoir que 1 et 1 seul "num_loc" pour 1 "numpr + numheure";
CT3: il ne peut y avoir que 1 et 1 seul "numsec" pour 1 "numet";
CT4: il ne peut y avoir que 1 et 1 seul "numop" pour 1 "numet + numheure";
CT5: il ne peut y avoir que 1 et 1 seul "numob" pour 1 "numsec + numheure";
CT6: il ne peut pas y avoir "numsec + numet + numheure" pour "numop + numob".

Messages de sortie:
sortie1: ensemble des valeurs d' attributs de ce TE.

Effets sur le S.I.:
Cette fonction n' introduit aucun changement dans la B.D.

Règles de traitements:
R1: liste des valeurs d' attributs de la valeur d' entité demandée.

Fonction: CONSULTATION

Objectifs:

A partir de n' importe quelle valeur de TE appartenant à la B.D., on souhaite obtenir les valeurs des autres TE appartenant à la B.D.

Messages en entrée:

numsec: numéro de la section.
numet: numéro de l' étudiant.
numob: numéro du cours du tronc commun.
numop: numéro du cours à option.
numpr: numéro du professeur.
num_loc: numéro du local.
numheure: numéro de l' heure.

Contraintes:

Les données vérifient les contraintes suivantes:

CT1: il ne peut y avoir que 1 et 1 seul "numob" ou"numop" pour 1 "numpr + numheure";
CT2: il ne peut y avoir que 1 et 1 seul "num_loc" pour 1 "numpr + numheure";
CT3: il ne peut y avoir que 1 et 1 seul "numsec" pour 1 "numet";
CT4: il ne peut y avoir que 1 et 1 seul "numop" pour 1 "numet + numheure";
CT5: il ne peut y avoir que 1 et 1 seul "numob" pour 1 "numsec + numheure";
CT6: il ne peut pas y avoir "numsec + numet + numheure" pour "numop + numob".

Messages de sortie:

sortie1: ensemble des relations répondant au message d' entrée.

Effets sur le S.I.:

Cette fonction n' introduit aucun changement dans la B.D.

Règles de traitements:

R1: imprimer sortie1.

Fonction: LISTE_VIDE

Objectifs:
Cette fonction permettra à l' utilisateur de prendre connnaissance des heures libres et des locaux libres pour 1 section/ 1 étudiant(e)/ 1 professeur.

Messages en entrée:
numsec: numéro de la section.
numet: numéro de l' étudiant.
numpr: numéro du professeur.
num_loc: numéro du local.

Contraintes:
Les données vérifient les contraintes suivantes:
CT1: il ne peut y avoir que 1 et 1 seul "num_loc" pour 1 "numpr + numheure";
CT2: il ne peut y avoir que 1 et 1 seul "numsec" pour 1 "numet";

Messages de sortie:
sortie1: ensemble des heures inoccupées soit pour 1 section soit pour 1 étudiant soit pour 1 professeur;
sortie2: ensemble des locaux vides à une heure déterminée.

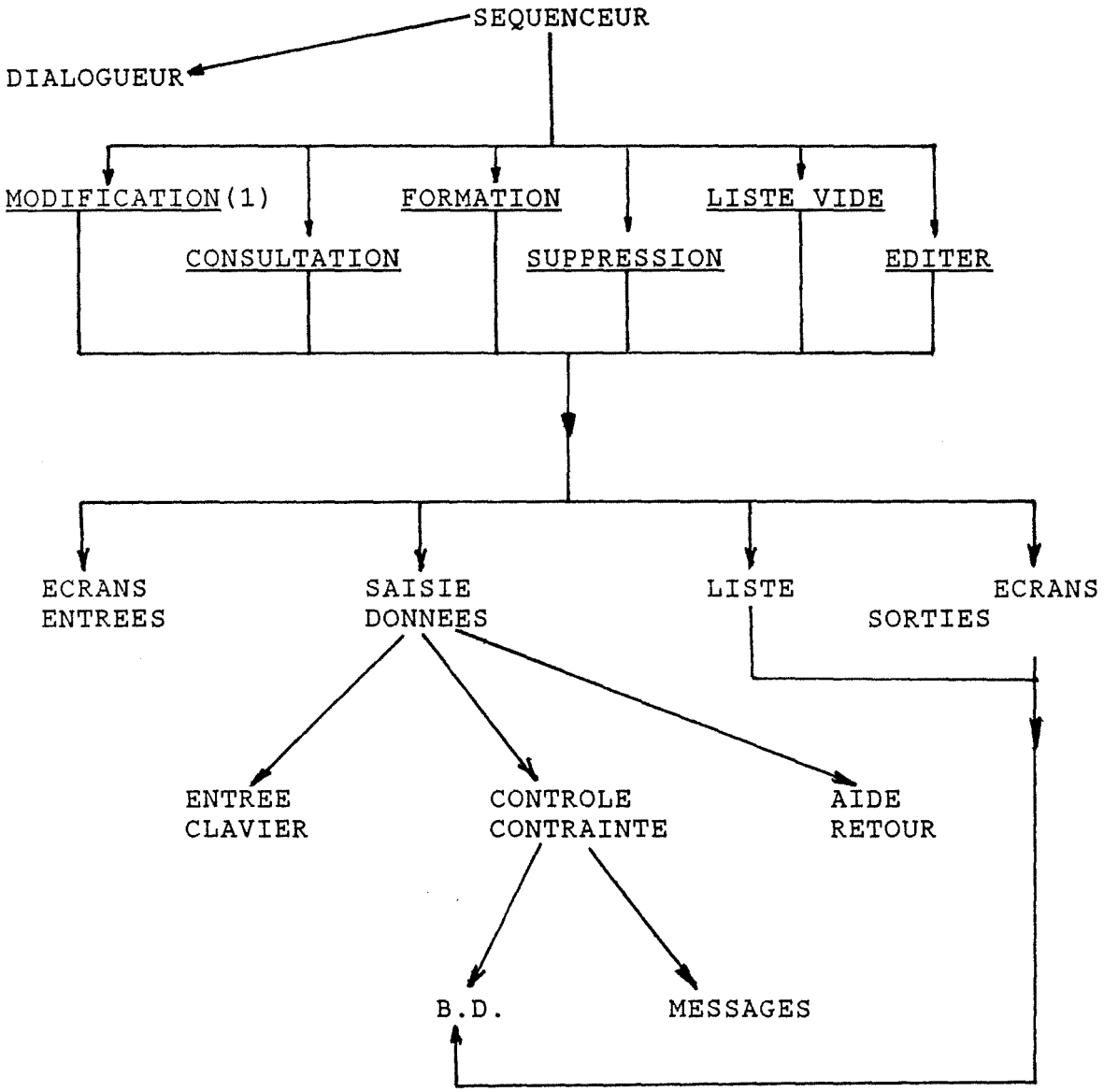
Effets sur le S.I.:
Cette fonction n' introduit aucun changement dans la B.D.

Règles de traitements:
R1: affichage des heures libres pour 1 section/ 1 étudiant(e)/ 1 professeur;
R2: affichage des locaux libres pour une heure déterminée.

3.3.Conception Globale.

3.3.1.Découpe en Modules Logiques.

1.Schéma de la Découpe en Modules.



(1)Les modules dont le nom est souligné sont des fonctions de l' analyse fonctionnelle.

2.Description Succincte du Rôle de chaque Module.

module SEQUENCEUR

Le module séquenceur est le module de coordination des différentes fonctions (au nombre de 6) qui sont le résultat de l' analyse fonctionnelle.

Son rôle est de permettre à l' utilisateur de choisir une des applications dans le menu.

Le menu est repris dans le dialogueur.

Lorsque l' utilisateur en a terminé avec son application, il est à nouveau en mesure d' en choisir une autre.

module DIALOGUEUR

Le module dialogeur reprend tous les menus dont a besoin un programme.

Le menu présenté est un menu déroulant répondant à certains critères ergonomiques.

module MODIFICATION

Provient de la fonction modification.

module SUPPRESSION

Provient de la fonction suppression.

module FORMATION

Provient de la fonction formation.

module CONSULTATION

Provient de la fonction consultation.

module LISTE_VIDE

Provient de la fonction liste-vide.

module EDITER

Provient de la fonction editer.

module LISTE

Toutes les fonctions de l' analyse fonctionnelle s' y réfèrent pour afficher les valeurs d' un T.E.

On fera des saisies dans tout ce qui concerne la consultation/editer/liste_vide.

Dans les autres fonctions, aucun traitement ne pourra être réalisé.

module ECRANS ENTREES

Présente, lors de l' application, le séquencement des écrans prêts à la saisie.

module SAISIE DONNEES

Remplit l' écran de données à partir de la B.D. suivant les primitives propres à la fonction en cours.

module ECRANS SORTIES

Répond aux questions posées par l' utilisateur (consultation, éditer, liste-vide) sur moniteur ou sur imprimante.

module ENTREE CLAVIER

L' utilisateur peut déplacer le curseur en pleine page, se positionner sur un élément et saisir cet élément.

module CONTROLE CONTRAINTE

Vérifie la validité de la saisie par rapport à des contraintes préalablement définies et en ayant accès à la B.D.

module AIDE RETOUR

Met à la disposition de l' utilisateur des écrans d' aide à l' utilisation et des écrans de repérage pour la poursuite de l' application.(1)

module B.D.

Contient uniquement des primitives d' accès et de création de chemin.

Ce module reçoit un entier déterminant la fonction appelée.

module MESSAGE

Reprend l' ensemble des messages dont le programme doit se servir.

Vu l' aspect ergonomique et de guidage, les seuls messages sont des messages d' erreur.

---

(1)La réalisation d' un horaire est un long travail, il demande une forte concentration que l' utilisateur n' est pas en mesure d' assurer.

3.3.2.Spécifications Externes.

module SEQUENCEUR

relation:

"utilise" dialogueur

"déclenche" modification
consultation
formation
suppression
liste_vide
editer

module DIALOGUEUR

arguments:

choix: caractère
sel: entier

pré-conditions:

choix prend les valeurs:

"M": pour la fonction modification,
"S": pour la fonction suppression,
"A": pour la fonction ajout,
"C": pour la fonction consultation,
"L": pour la fonction liste_vide,
"E": pour la fonction éditer,
"Q": pour quitter.

sel prend les valeurs:

"5": pour les professeurs,
"6": pour les étudiants,
"7": pour les sections,
"8": pour les cours à option,
"9": pour les cours obligatoires,
"10":pour les locaux,
"11":quitter.

résultat:

sélection dans un menu déroulant de la fonction et de l' entité d' introduction de la logique d' utilisation.

module MODIFICATION / module AJOUT / module SUPPRESSION

argument:

sel: entier

pré-condition:

sel prend les valeurs:

"5": pour les professeurs,
"6": pour les étudiants,
"7": pour les sections,
"8": pour les cours à option,
"9": pour les cours obligatoires,
"10":pour les locaux,
"11":quitter.

résultats:

ecran: entier
prim: entier
action: entier
typo: entier

post-conditions:

ecran prend les valeurs:

"1" pour écran professeur,
"2" pour écran étudiant,
"3" pour écran section,
"4" pour écran cours à option,
"5" pour écran cours obligatoire,
"6" pour écran local.

typo prend les valeurs:

"1" pour la partie suppression de la modification,
"2" pour la partie ajout de la modification,
"0" pour les autres cas.

prim prend les valeurs:

"1" lister professeur,
"2" lister étudiant,
"3" lister section,
"4" lister option,
"5" lister obligatoire,
"6" lister local.

action prend les valeurs:

"2" créer section-obligatoire,
"3" créer étudiant-option,
"4" créer obligatoire-heure,

```
                    "5" créer option-heure,
                    "6" créer obligatoire-local,
                    "7" créer option-local,
                    "8" créer obligatoire-professeur,
                    "9" créer option-professeur,
                    "10" scanner section-obligatoire,
                    "11" scanner étudiant-option,
                    "12" scanner obligatoire-local,
                    "13" scanner option-local,
                    "14" scanner obligatoire-professeur,
                    "15" scanner option-professeur,
                    "16" scanner obligatoire-heure,
                    "17" scanner option-heure,
                    "18" scanner section-étudiant,
                    "19" supprimer section-obligatoire,
                    "20" supprimer étudiant-option,
                    "21" supprimer obligatoire-local,
                    "22" supprimer option-local,
                    "23" supprimer obligatoire-professeur,
                    "24" supprimer option-professeur,
                    "25" supprimer section-étudiant,
                    "26" supprimer obligatoire-heure,
                    "27" supprimer option-heure,
                    "28" scanner professeur-obligatoire,
                    "29" scanner professeur-option,
                    "30" scanner option-étudiant,
                    "31" scanner obligatoire- section,
                    "32" scanner local-obligatoire,
                    "33" scanner local-option.

    relation:

        "utilise" ecrans entrees,
                  saisie donnees,
                  ecrans sorties,
                  liste.

module EDITER

    argument:

        sel: entier

    pré-condition:

        sel prend les valeurs:

                    "5": pour les professeurs,
                    "6": pour les étudiants,
                    "7": pour les sections,
                    "8": pour les cours à option,
                    "9": pour les cours obligatoires,
                    "10":pour les locaux,
                    "11":quitter.

    résultat:

        prim: entier
```

post-condition:

prim prend les valeurs:

"1" lister professeur,
"2" lister étudiant,
"3" lister section,
"4" lister option,
"5" lister obligatoire,
"6" lister local.

relation:

"utilise" ecrans entrees,
saisie donnees,
ecrans sorties.

module CONSULTATION

argument:

sel: entier

pré-condition:

sel prend les valeurs:

"5": pour les professeurs,
"6": pour les étudiants,
"7": pour les sections,
"8": pour les cours à option,
"9": pour les cours obligatoires,
"10":pour les locaux,
"11":quitter.

résultats:

ecran: entier
prim: entier
action: entier

post-condition:

ecran prend les valeurs:

"1" pour écran professeur,
"2" pour écran étudiant,
"3" pour écran section,
"4" pour écran cours à option,
"5" pour écran cours obligatoire,
"6" pour écran local.

prim prend les valeurs:

"1" lister professeur,
"2" lister étudiant,
"3" lister section,
"4" lister option,

```
                    "5" lister obligatoire,
                    "6" lister local.
          action prend les valeurs:

                    "10" scanner section-obligatoire,
                    "11" scanner étudiant-option,
                    "12" scanner obligatoire-local,
                    "13" scanner option-local,
                    "14" scanner obligatoire-professeur,
                    "15" scanner option-professeur,
                    "16" scanner obligatoire-heure,
                    "17" scanner option-heure,
                    "18" scanner section-étudiant,
                    "28" scanner professeur-obligatoire,
                    "29" scanner professeur-option,
                    "30" scanner option-étudiant,
                    "31" scanner obligatoire- section,
                    "32" scanner local-obligatoire,
                    "33" scanner local-option.

     relation:

          "utilise" ecrans entrees,
                    saisie donnees,
                    ecrans sorties,
                    liste.

module LISTE_VIDE

     argument:

          sel: entier

     pré-condition:

          sel prend les valeurs:

                    "5": pour les professeurs,
                    "6": pour les étudiants,
                    "7": pour les sections,
                    "8": pour les cours à option,
                    "9": quitter.

     résultats:

          ecran:  entier
          prim:   entier
          action: entier

     post-conditions:

          ecran prend les valeurs:

                    "1" pour écran professeur,
                    "2" pour écran étudiant,
                    "3" pour écran section,
                    "6" pour écran local.
```

prim prend les valeurs:

"1" lister professeur,
"3" lister section,
"6" lister local.

action prend les valeurs:

"10" scanner section-obligatoire,
"11" scanner etudiant-option,
"12" scanner obligatoire-local,
"13" scanner option-local,
"14" scanner obligatoire-professeur,
"15" scanner option-professeur,
"16" scanner obligatoire-heure,
"17" scanner option-heure,
"18" scanner section-étudiant,
"28" scanner professeur-obligatoire,
"29" scanner professeur-option,
"30" scanner option-étudiant,
"31" scanner obligatoire- section,
"32" scanner local-obligatoire,
"33" scanner local-option.

relation:

"utilise" ecrans entrees,
saisie donnees,
ecrans sorties,
liste.

module LISTE

argument:

prim: entier

pré-condition:

prim prend les valeurs:

"1" lister professeur,
"2" lister étudiant,
"3" lister section,
"4" lister option,
"5" lister obligatoire,
"6" lister local.

résultat:

liste des éléments appartenant à la B.D.

module ECRANS ENTREES

arguments:

ecran: entier
typo: entier

choix: caractère

pré-conditions:

ecran prend les valeurs:

"1" pour écran professeur,
"2" pour écran étudiant,
"3" pour écran section,
"4" pour écran cours à option,
"5" pour écran cours obligatoire,
"6" pour écran local.

typo prend les valeurs:

"1" pour la partie suppression de la modification,
"2" pour la partie ajout de la modification,
"0" pour les autres cas.

choix prend les valeurs:

"M": pour la fonction modification,
"S": pour la fonction suppression,
"A": pour la fonction ajout,
"C": pour la fonction consultation,
"L": pour la fonction liste_vide,
"E": pour la fonction éditer,
"Q": pour quitter.

résultat:

affichage du bon écran au bon moment.

module SAISIE DONNEES

arguments:

ecran: entier
prim: entier
action: entier

pré-conditions:

ecran prend les valeurs:

"1" pour écran professeur,
"2" pour écran étudiant,
"3" pour écran section,
"4" pour écran cours à option,
"5" pour écran cours obligatoire,
"6" pour écran local.

prim prend les valeurs:

"1" lister professeur,
"2" lister étudiant,
"3" lister section,
"4" lister option,
"5" lister obligatoire,
"6" lister local.

action prend les valeurs:

"2" créer section-obligatoire,
"3" créer étudiant-option,
"4" créer obligatoire-heure,
"5" créer option-heure,
"6" créer obligatoire-local,
"7" créer option-local,
"8" créer obligatoire-professeur,
"9" créer option-professeur,
"10" scanner section-obligatoire,
"11" scanner étudiant-option,
"12" scanner obligatoire-local,
"13" scanner option-local,
"14" scanner obligatoire-professeur,
"15" scanner option-professeur,
"16" scanner obligatoire-heure,
"17" scanner option-heure,
"18" scanner section-étudiant,
"19" supprimer section-obligatoire,
"20" supprimer étudiant-option,
"21" supprimer obligatoire-local,
"22" supprimer option-local,
"23" supprimer obligatoire-professeur,
"24" supprimer option-professeur,
"25" supprimer section-étudiant,
"26" supprimer obligatoire-heure,
"27" supprimer option-heure,
"28" scanner professeur-obligatoire,
"29" scanner professeur-option,
"30" scanner option-étudiant,
"31" scanner obligatoire- section,
"32" scanner local-obligatoire,
"33" scanner local-option.

résultat:

suivant la fonction qui a été sélectionnée préalablement, les données sont traitées à l' aide de primitives de la B.D.

relation:

"utilise": entree clavier
contrôle contrainte
aide retour

module ECRAN SORTIE

arguments:

ecran: entier
periph: entier
choix: caractère

pré-condition:

ecran prend les valeurs:

"1" pour écran professeur,
"2" pour écran étudiant,
"3" pour écran section,
"4" pour écran cours à option,
"5" pour écran cours obligatoire,
"6" pour écran local.

periph prend les valeurs:

"1" pour écran
"2" pour imprimante

choix prend les valeurs:

"M": pour la fonction modification,
"S": pour la fonction suppression,
"A": pour la fonction ajout,
"C": pour la fonction consultation,
"L": pour la fonction liste_vide,
"E": pour la fonction éditer,
"Q": pour quitter.

résultat:

affichage sur le périphérique désiré.

relation:

"utilise" B.D.

module ENTREE CLAVIER

argument:

position du curseur suivant l' axe X et l' axe Y.

résultat:

numero: entier

post-condition:

numero = position du curseur suivant l' axe Y
+ 15 * (position du curseur suivant l' axe X
div 20)

module CONTROLE CONTRAINTE

arguments:

num: entier
ecran: entier

pré-conditions:

num prend les valeurs:

1 à 45

ecran prend les valeurs:

"1" pour écran professeur,
"2" pour écran étudiant,
"3" pour écran section,
"4" pour écran cours à option,
"5" pour écran cours obligatoire,
"6" pour écran local.

résultats:

prim: entier
action: entier
nmess: entier

post-conditions:

prim prend les valeurs:

"1" lister professeur,
"2" lister étudiant,
"3" lister section,
"4" lister option,
"5" lister obligatoire,
"6" lister local.

action prend les valeurs:

"10" scanner section-obligatoire,
"11" scanner étudiant-option,
"12" scanner obligatoire-local,
"13" scanner option-local,
"14" scanner obligatoire-professeur,
"15" scanner option-professeur,
"16" scanner obligatoire-heure,
"17" scanner option-heure,
"18" scanner section-étudiant,
"28" scanner professeur-obligatoire,
"29" scanner professeur-option,
"30" scanner option-étudiant,
"31" scanner obligatoire- section,
"32" scanner local-obligatoire,
"33" scanner local-option.

nmess correspond aux messages suivant:

1. CLASSE OCCUPEE
2. ETUDIANT(E) OCCUPE(E)
3. TOUTES les HEURES de ce COURS sont PLACEES
4. PROFESSEUR OCCUPE
5. LOCAL OCCUPE
6. PROFESSEUR INDISPONIBLE
7. LOCAL INDISPONIBLE
8. TOUS les LOCAUX sont OCCUPES à cet instant
9. TOU(TE)S les ETUDIANT(E)S sont OCCUPE(E)S à cet instant
10. TOUTES les SECTIONS sont OCCUPEES à cet instant
11. TOUS les PROFESSEURS sont OCCUPES à cet instant
12. TOUS les PROFESSEURS sont INDISPONIBLES à cet instant
13. TOUS les LOCAUX sont INDISPONIBLES à cet instant
14. L' HORAIRE de cette SECTION est COMPLET
15. L' HORAIRE de cet ETUDIANT(E) est COMPLET
16. L' HORAIRE de ce PROFESSEUR est COMPLET

relation:

"utilise" message.

module AIDE RETOUR

arguments:

F1: touche fonction
ecran: entier
choix: caractère

pré-conditions:

choix prend les valeurs:

"M": pour la fonction modification,
"S": pour la fonction suppression,
"A": pour la fonction ajout,
"C": pour la fonction consultation,
"L": pour la fonction liste_vide,
"E": pour la fonction éditer,
"Q": pour quitter.

ecran prend les valeurs:

"1" pour écran professeur,
"2" pour écran étudiant,
"3" pour écran section,
"4" pour écran cours à option,
"5" pour écran cours obligatoire,
"6" pour écran local.

résultats:

affichage d' écrans d' aide et de rappel des données entrées.

post-conditions:

ces écrans reprennent

* un descriptif de ce qu' on peut faire,
* un descriptif de ce qui vient d' être fait:
numet, numpr, numsec, numloc, numop, numob.

module B.D.

arguments:

prim: entier
action: entier
numero: entier
numloc: type X999

pré-conditions:

prim prend les valeurs:

"1" lister professeur,
"2" lister étudiant,
"3" lister section,
"4" lister option,
"5" lister obligatoire,
"6" lister local.

numero = position du curseur suivant l' axe Y
+ 15 * (position du curseur suivant l' axe X
div 20)

action prend les valeurs:

"10" scanner section-obligatoire,
"11" scanner étudiant-option,
"12" scanner obligatoire-local,
"13" scanner option-local,
"14" scanner obligatoire-professeur,
"15" scanner option-professeur,
"16" scanner obligatoire-heure,
"17" scanner option-heure,
"18" scanner section-étudiant,
"28" scanner professeur-obligatoire,
"29" scanner professeur-option,
"30" scanner option-étudiant,
"31" scanner obligatoire- section,
"32" scanner local-obligatoire,
"33" scanner local-option.

résultat:

dbstatus: entier
numpr: entier
numet: entier
numsec: entier
numop: entier
numob: entier

numloc: type X999.

post-condition:

voir return codes (1)

module MESSAGES

argument:

nmess: entier

pré-condition:

nmess prend les valeurs:

1 à 16

résultat:

message correspondant au numéro d' entrée

post-condition:

les messages sont:

1. SECTION OCCUPEE
2. ETUDIANT(E) OCCUPE(E)
3. TOUTES les HEURES de ce COURS sont PLACEES
4. PROFESSEUR OCCUPE
5. LOCAL OCCUPE
6. PROFESSEUR INDISPONIBLE
7. LOCAL INDISPONIBLE
8. TOUS les LOCAUX sont OCCUPES à cet instant
9. TOU(TE)S les ETUDIANT(E)S sont OCCUPE(E)S à cet instant
10. TOUTES les SECTIONS sont OCCUPEES à cet instant
11. TOUS les PROFESSEURS sont OCCUPES à cet instant
12. TOUS les PROFESSEURS sont INDISPONIBLES à cet instant
13. TOUS les LOCAUX sont INDISPONIBLES à cet instant
14. L' HORAIRE de cette SECTION est COMPLET
15. L' HORAIRE de cet ETUDIANT(E) est COMPLET
16. L' HORAIRE de ce PROFESSEUR est COMPLET
17. La TAILLE de la SECTION est SUPERIEURE à la TAILLE du LOCAL
18. LOCAL mal ADAPTE aux ACTIVITES
19. ACTIVITE mal ADAPTEE au LOCAL
20. SATURATION

(1) [3.1.] The Basic Programming Interface

3.4.Interface.

3.4.1.Introduction.

Nous distinguons trois types d'écrans:

- Menu déroulant:

  Cet écran correspond au module dialogueur,

- Ecrans de saisies de données ou de consultation:

  Ces écrans correspondent au module ecrans entrees,

- Grille horaire:

  Cet écran correspond au module ecrans sorties.

Sur chacun de ces écrans, peuvent apparaître des fenêtres de messages ou d' aides.

Nous privilégions les choix faits à l' aide de flèches au rentrée par le clavier de valeurs.

3.4.2.Menu Déroulant.

Un menu a pour principe de proposer des choix à l' utilisateur.

Différents types de menus existent; entre autres:

- sélection du premier caractère lettre/chiffre sur le clavier alphanumérique,
- sélection à l' aide des touche "flèche" et "return".

Notre menu reprend les fonctions de l' analyse fonctionnelle sur un bandeau où se balade le curseur et représente le niveau de ralliement; c' est à ce niveau qu' on se retrouve à la fin de chaque opération.

Dans le déroulement, sous chaque fonction du bandeau, on a des sous-choix permettant un guidage pour différentes logiques d' utilisation.

Ces sous-choix sont les entités définies dans le schéma conceptuel:

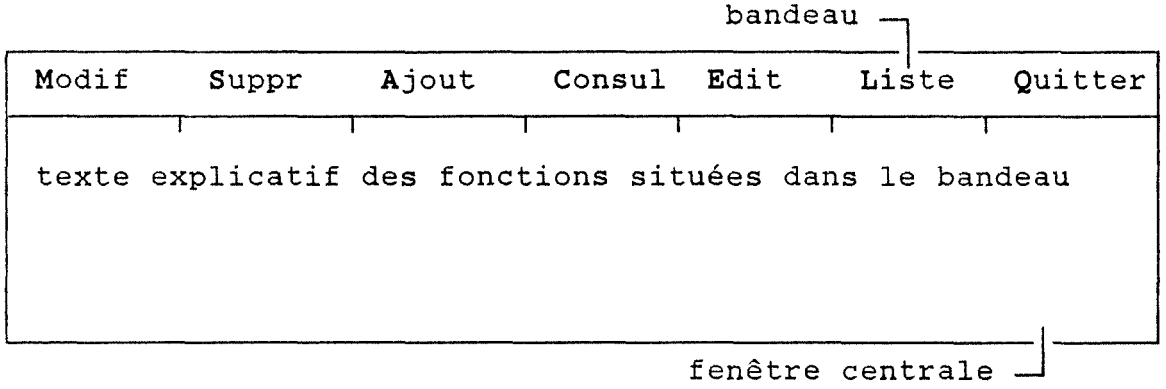
- professeur,
- étudiant(e),
- section,
- local,
- option,
- obligatoire.

A tous les niveaux de menu, il faut une sortie vers le niveau supérieur:

- niveau 1: (niveau du bandeau)

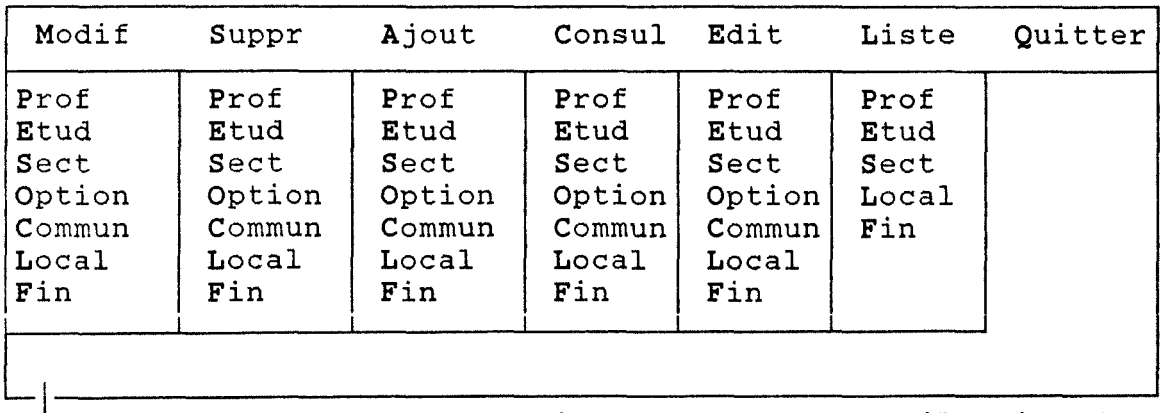
  modification (Modif)
  suppression (Suppr)
  ajout (Ajout)
  consultation (Consul)
  editer (Edit)
  liste_vide (Liste)
  quitter (vers le DOS) (Quitter)

  écran de niveau 1:



- niveau 2: (niveau déroulement)

  professeur
  etudiant(e)
  section
  option
  obligatoire
  local
  fin (retour vers le niveau 1)



fenêtre déroulante apparaissant lors de la sélection de modification au niveau 1

Une seule de ces six fenêtres apparaît après la sélection de niveau 1.

Ce deuxième niveau n' apparaît pas à tous les moments; on veille à ce que ne soit affiché que le niveau nécessaire ou à ce que le déplacement du curseur soit limité aux zones utiles.

Le curseur à ce niveau est un bandeau de couleur tranchante mettant en évidence la sélection de l' élément.

Sur le centre du moniteur, on inscrit les renseignements utiles à l' utilisation du logiciel.

Remarques:

Nous avons décidé:

- de permettre la sélection soit par le déplacement des flèches soit par l' introduction de la première lettre de la fonction choisie (de préférence en surbrillance).

- de permettre le déplacement des flèches non pas à l' aide de la souris mais des touches "flèches".

3.4.3.Ecrans de Saisies de Données ou de Consultation.

Ces écrans apparaissent après le menu déroulant.

Deux types de travaux sur ces écrans:

- affichage de ces écrans,

- entrée des données.

1.Affichage.

Les fonctions consultation, editer, liste_vide ne permettent que ce niveau: en effet, aucune saisie ne doit être réalisée.

A la fin d' une opération de ce niveau, deux possibilités s' offrent:

- retour au menu principal,
- entrée des données.

Dans le premier cas, le retour se fait en actionnant la touche "return".

Dans le second cas, on passe à l' étape entrée des données.

Le séquencement des écrans intervient dans la logique d' utilisation.

Notre séquencement est très serré et ne laisse aucune liberté à l' utilisateur si ce n' est le choix initial de la fonction et du premier élément à introduire (cf. 3.4.2.).

Le séquencement suit le schéma suivant:

- par professeur:

    affichage des écrans:
    professeur,
    obligatoire,
    option,
    section,
    (étudiant(e)), (dans le cas de cours à option)
    local,
    suivant cet ordre,

- par étudiant:

    affichage des écrans:
    section,
    étudiant(e),
    option,
    professeur,
    local,
    suivant cet ordre.

- par section:

    affichage des écrans:
    section,
    obligatoire,
    professeur,
    local,
    suivant cet ordre.

- par option:

    affichage des écrans:
    option,
    professeur,
    section,
    étudiant(e),
    local,
    suivant cet ordre.

- par obligatoire:

    affichage des écrans:
    obligatoire,
    professeur,
    section,
    local,
    suivant cet ordre.

- par local:

affichage des écrans:
local,
obligatoire,
option,
section,
(étudiant(e)), (dans le cas de cours à option)
professeur,
suivant cet ordre.

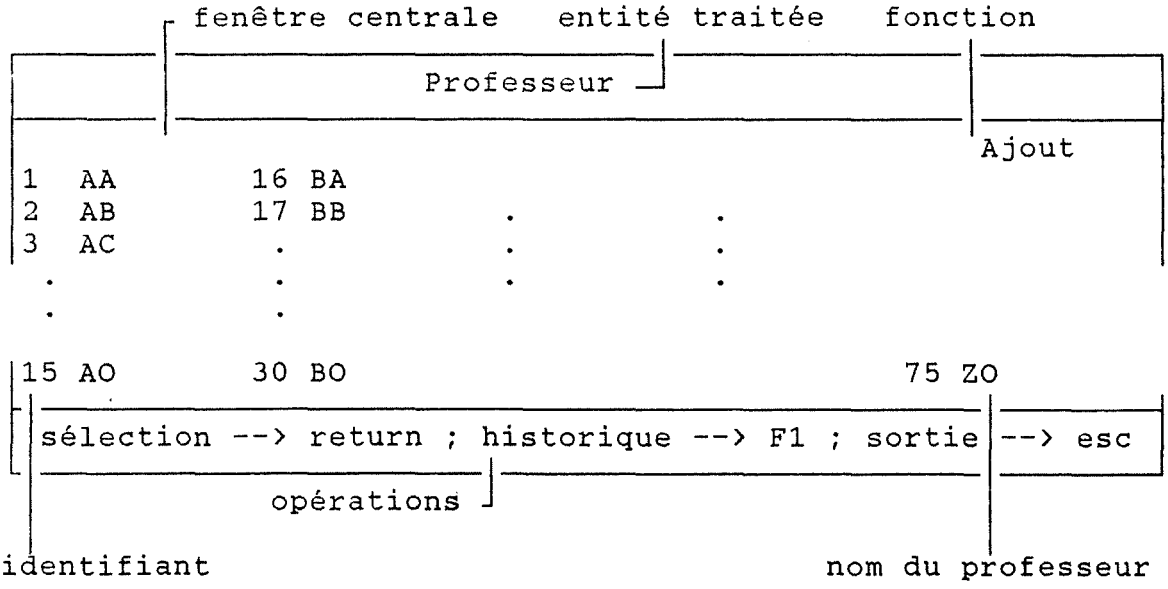
Le curseur est inefficace.

A ce niveau, les fenêtres d' aide traitent du réaffichage des données préalablement mémorisées pour ce problème et permettent une interruption volontaire ou involontaire de la part de l' utilisateur.

Exemple:

Nous avons choisi d' ajouter un élément d' horaire en commençant par le professeur.

L' écran professeur apparaît:



2.Entrée des Données.

Les données saisies sont choisies parmi un lot de valeurs d' entités appartenant à la B.D. préalablement remplie (pour éviter des problèmes de cohérence).(1)

---

(1) Les mises à jour ne portent que sur les "chemins". En effet, nous considérons que les attributs de tous les T.E. ont été préalablement affectées par l' utilisateur. Il semble dangereux de laisser des novices jouer avec l' intégrité de la B.D.

Le but de ce logiciel est de créer ou supprimer des chemins entre deux types d' entités.

Il est nécessaire de saisir les deux valeurs d' entités avant de créer ou supprimer quoi que ce soit.

Il faut noter que la saisie se fait dans le sens opposé au traitement.

NB: saisie
Lors de la saisie, l' utilisateur choisi d' abord le type d' action (suppression, modification, ...) et puis sélectionne les objets sur lesquels sera appliquée l' action.

traitement
Lors du traitement, il faut attendre que soient entrés les deux objets pour pouvoir les traiter

On évite de demander à l' utilisateur le même renseignement plusieurs fois lors de la réalisation d' un élément de l' horaire.

La saisie se fait par le curseur sous forme de bandeau.

Le choix se fait par déplacement à l' aide des flèches du bandeau et réalisation de la sélection par la touche "return".

Après la sélection, une fenêtre apparaît proposant une alternative:

- OK, (F10)
- à reconsidérer. (escape)

Pour passer à l' écran suivant et confirmer de façon irrémédiable son choix (OK), on appuie sur F10.

Dans l' autre cas, deux possibilités se présentent:

- on appuie sur "escape" et on refait son choix,
- une autre fenêtre s' ouvre annonçant l' impossibilité de satisfaire la demande; on appuie alors sur "escape" et on répète son choix.

Au passage à l' écran suivant peuvent se présenter trois types d' écrans:

- Menu déroulant,
- Affichage,
- Grille horaire.

Le déplacement du curseur doit se limiter aux zones où il y a des valeurs.

Le classement des valeurs d' une entité est fait en fonction de son numéro identifiant.

Cela semble un choix discutable dans certains cas et pas dans d' autres.

En effet, en ce qui concerne les locaux, les cours à option ou obligatoires, les sections, il n' y a pas de difficultés puisque, logiquement, l' utilisateur aura entré ses valeurs de façon groupée.(1)

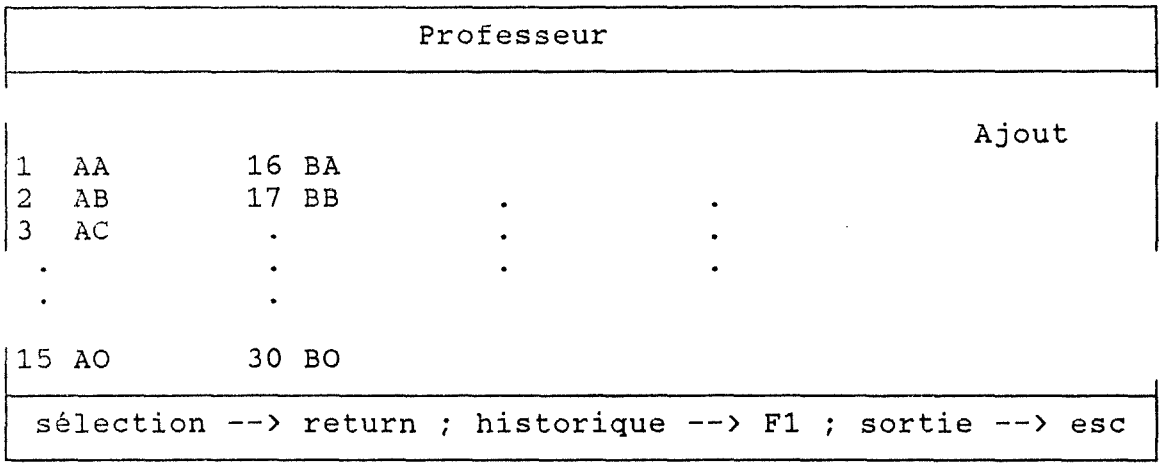
D' un autre côté, les étudiants et les professeurs peuvent être triés soit d' après leur numéro identifiant soit d' après leur nom et prénom.

A ce niveau, les fenêtres d' aide traitent de l' apparition des messages d' erreurs commises lors de saisies ou lorsque l' horaire pour cette entité est en voie d' être complètée.

Exemple:

Nous avons choisi d' ajouter un élément d' horaire en commençant par le professeur.

L' écran professeur apparaît:

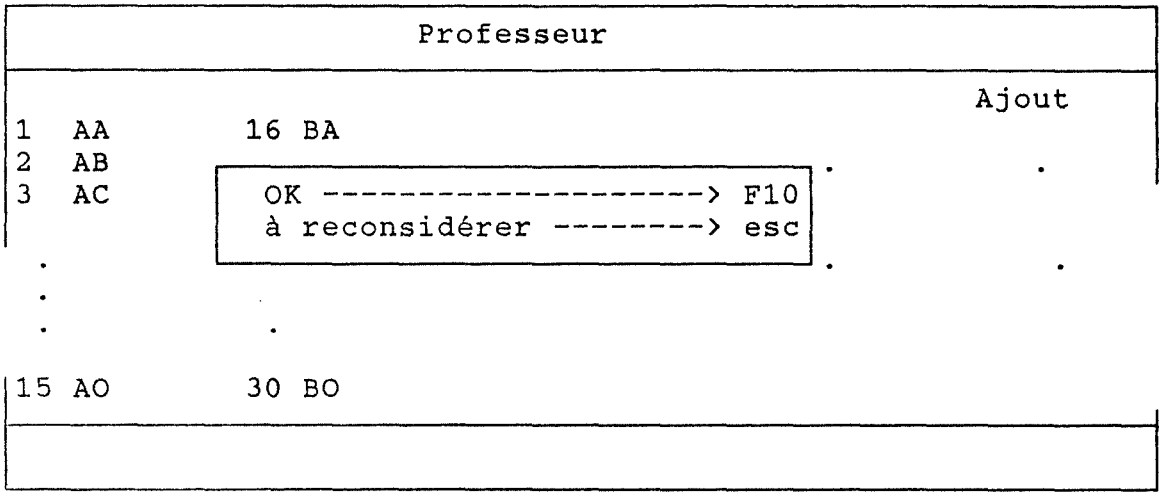


Le curseur se déplace sur la fenêtre centrale et uniquement où des noms de professeurs sont affichés.

L' utilisateur appuie sur "return".

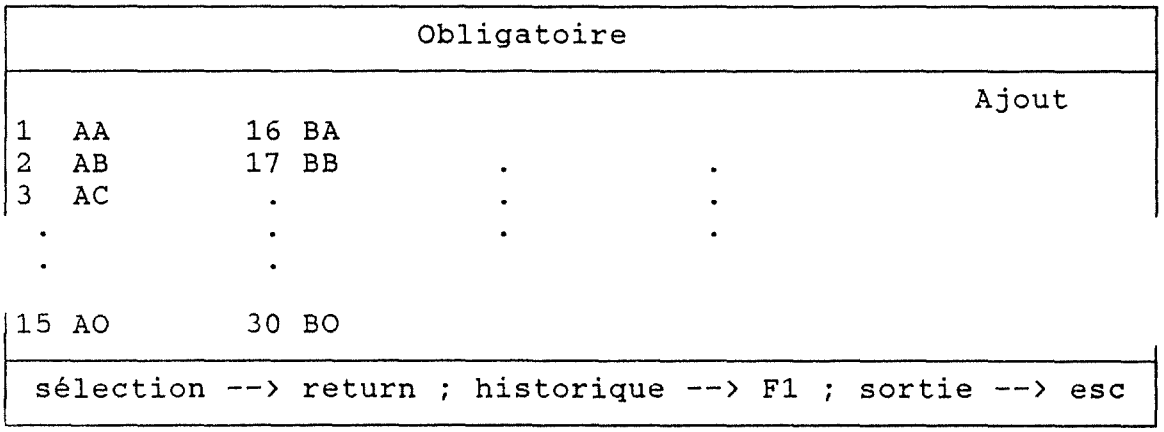
(1)Tous les cours seront par exemple regroupés comme suit:
- math 4 : 1
- math 7 : 2
- latin 4 : 3
- latin 5 : 4

La fenêtre suivante apparaît sur cet écran:



Si on reconsidère, on retombe sur l' écran professeur.

Si c' est OK, apparaît l' écran obligatoire:

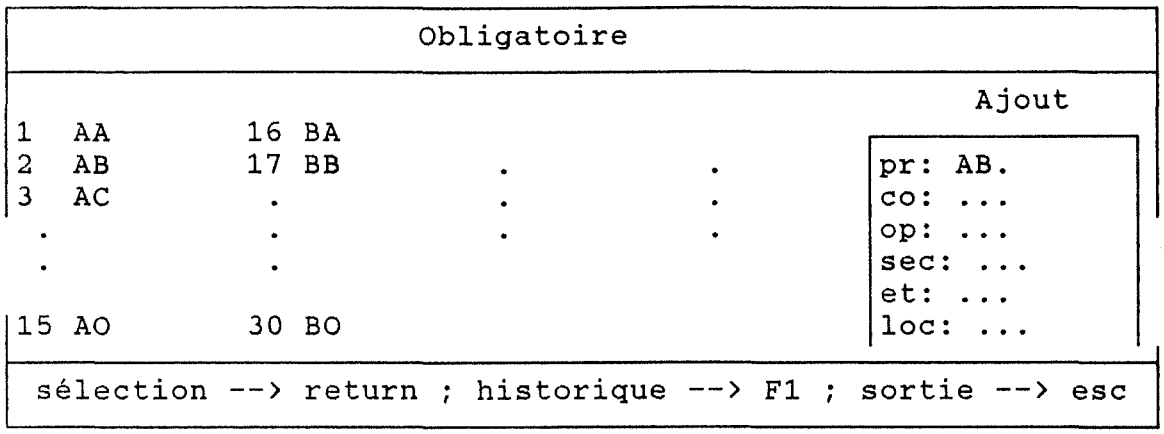


On remplace le nom des professeur par le nom des cours.

Pour effectuer une sélection, on opère de la même façon que précédemment.

Dans le cas où l' utilisateur s' est interrompu pour des raisons diverses, il peut appeler une fenêtre lui rappelant les données déja entrée pour cet élément d' horaire en appuyant sur historique (F1).

Voici l' écran après sélection de la touche F1:



Si nous reprenons en détails cette mini fenêtre, nous avons:

pr: < numéro du professeur > < abrégé du nom >,
co: < numéro du cours commun > < abrégé de l' intitulé >,
op: < numéro du cours à option > < abrégé de l' intitulé >,
sec: < numéro de la section > < nombre d' étudiants >,
et: < numero d' étudiant > < abrégé du nom >,
loc: < numéro du local > < contenance maximum du local >.

Toutes ces rubriques ne sont pas toujours remplies, dans notre cas, seul "pr" est rempli.

Le curseur se déplace uniquement dans la fenêtre centrale.

3.4.4.Grille Horaire.

Une façon simple de représenter un horaire (quel que soit le domaine) reste une représentation graphique regroupant le plus d' information possible.

	1	2	3	4	5	6	7	8
lun								
mar								
mer								
jeu								
ven								

La grille horaire semble être la bonne représentation; elle s' applique bien à une lecture sur moniteur et sur papier.

De plus, il s' agit d' un formalisme habituel pour l' utilisateur.

Un élément déterminant pour cette grille est le cycle de l' activité: dans notre cas la semaine de cinq jours.

Nous découpons cette grille en:

- cinq jours,
- huit heures.

Cette grille peut apparaître en un seul écran.

Cette grille peut contenir des informations pour tout un horaire ou pour seulement un professeur, une section, un étudiant, un local.

Au moment de l' utilisation de la grille, tous les autres chemins entre les autres T.E. de cet élément d' horaire sont connus et avec leurs contraintes.

Exemple:

On a créé les chemins d' un élément d' horaire de ce type:

- professeur: AA,
- section: 11aA,
- cours obligatoire: math,
- local: 101.

A ce moment apparaît la grille horaire montrant les indisponibilités de professeurs, les indisponibilités de locaux.

Les valeurs devant être présentes lors de consultation sur cette grille sont, outre les coordonnées:

- nom du professeur (éventuellement en abrégé),
- numéro de(s) section(s),
- numéro du local,
- nom du cours à option ou obligatoire,
- les contraintes d' indisponibilité (sous forme de hachures) dans des cas bien définis (grille horaire d' un professeur,...).

Lors de saisies des heures, le déplacement se fait à l' aide des flèches et ne permet le déplacement que de case en case (idéalement en empêchant les zones hachurées).

A ce niveau, les fenêtres d' aide traitent de l' apparition des messages d' erreurs commises lors de la saisie des heures.(1)

A la fin de cette opération, le passage à l' écran suivant se fait par la touche F10.

Exemple1:

Ce type d' écran est celui apparaissant lors de mises à jour dans la B.D.

Prenons l' exemple d' une mini grille horaire se présentant lors de la création d' un horaire.

On a créé les chemins d' un élément d' horaire de ce type:

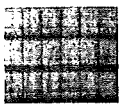
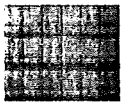
- professeur: AA,
- section: 11aA,
- cours obligatoire: math,
- local: A102.

Il reste à l' utilisateur le choix d' une heure de cours.

Il peut déplacer le curseur de case en case à l' aide des flèches.

Le choix de l' heure se faisant par la touche "return".

(1)Dans le cas où on peut aller sur des zones hachurées.

	1	2	3	4	5	6	7	8
lun								
mar				1LaA math A102				
mer			1LaA math A102					
jeu								
ven								

légende:

 = heure indisponible pour le professeur

 = local indisponible

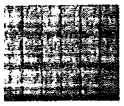
Exemple2:

Ce type d' écran est celui apparaissant lors de consultation de la B.D.

Dans ce cas, la grille peut être beaucoup plus remplie.

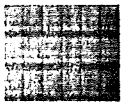
Prenons le cas de consultation, regardons où/quand/quoi/ à qui le professeur AA donne cours .

Le curseur est inefficace.

	1	2	3	4	5	6	7	8
lun								
mar				1LaA math A102	1LaC math A402			
mer			1LaA math A203					
jeu				3MA math A104				
ven								

légende:

Dans ce cas nous nous contenterons des contraintes sur l' entité désirée (professeur AA).

 = heure indisponible pour le professeur

3.5.Bibliographie.

[3.1.] J.-L. HAINAUT

NDBS: A Simple Data Base System for Small Computer

Institut d' Informatique des Facultés Universitaires de Namur - 1987

[3.2.] J.-J. MEYER

PRATIQUE du TURBO PASCAL Créez Vos Progiciels

Editions Radio

[3.3.] A. Van LAMSWEERDE

Cours de Méthodologie de Développement de gros Logiciels

Institut d' Informatique des Facultés Universitaires de Namur - 1989

[3.4.] D. SCAPIN

Guide Ergonomique de Conception des Interfaces Homme-Ordinateur

rapport INRIA - 1977

TRAITEMENT de la B.D.

4.1.Stockage et Traitement des Données.

Comme indiqué dans l' introduction de ce présent travail, un horaire met en oeuvre un très grand nombre de données.

Pour stocker ces données, nous allons utiliser une technique qui se révèle très pratique pour l' écriture de primitives d' accès et de mise à jour (1): la **Base de Données** (B.D.).

Le choix d' une B.D. s' est porté sur N.D.B.S., développé par les Facultés Notre Dame de la Paix, qui s' appuie sur le schéma E/A utilisé dans le chapitre 3.

La B.D. est utilisée non seulement pour les primitives des modules provenant de l' analyse fonctionnelle mais aussi pour les modules de vérification des contraintes.

Reprenons les différents modules provenant des fonctions de l' analyse fonctionnelle et décrivons leurs suites de primitives:

4.1.1.Fonctions de Mise à Jour de la B.D.

Nous écrivons en caractères gras les arguments d' interface à introduire par l' utilisateur.

Fonction: FORMATION

L' élément à introduire est:

Professeur

lister professeur
lister obligatoire
créer un chemin entre **professeur** ---> **obligatoire**
lister section
créer un chemin entre obligatoire --> **section**
lister local
créer un chemin entre obligatoire --> **local**
créer un chemin entre obligatoire --> **heure**

---

(1) Les mises à jour ne portent que sur les "chemins". En effet, nous considérons que les attributs de tous les T.E. ont été préalablement affectés par l' utilisateur. Il semble dangereux de laisser des novices jouer sur l' intégrité de la B.D.
ex: lister un chemin qui a été créé à partir d' une valeur d' identifiant inexistant.

ou

lister professeur
lister option
créer un chemin entre **professeur** ---> **option**
lister section
scanner chemin **section** ----------> étudiant(e) *
créer un chemin entre option -------> **étudiant(e)**
lister local
créer un chemin entre option -------> **local**
créer un chemin entre option -------> **heure**

Etudiant(e)

lister section
scanner chemin **section** ----------> étudiant(e) *
lister option
créer chemin entre **étudiant(e)** -----> **option**
lister professeur
créer chemin entre option ----------> **professeur**
lister local
créer chemin entre option ----------> **local**
créer chemin entre option ----------> **heure**

Section

lister section
lister obligatoire
créer un chemin entre **section** ------> **obligatoire**
lister professeur
créer un chemin entre obligatoire --> **professeur**
lister local
créer un chemin entre obligatoire --> **local**
créer un chemin entre obligatoire --> **heure**

Fonction: SUPPRESSION

L' élément à introduire est:

Professeur

lister professeur
scanner chemin **professeur** ----------> obligatoire
supprimer un chemin entre professeur ---> **obligatoire**
scanner chemin **obligatoire** ----------> heure
supprimer un chemin entre obligatoire --> **heure**
scanner chemin **obligatoire** ----------> section
supprimer un chemin entre obligatoire --> **section**
scanner chemin **obligatoire** ----------> local
supprimer un chemin entre obligatoire --> **local**

ou

lister professeur
scanner chemin **professeur** ----------> option
supprimer un chemin entre professeur ---> **option**
scanner chemin **option** ----------> heure
supprimer un chemin entre option -------> **heure**
scanner chemin **option** ----------> étudiant(e) *
supprimer un chemin entre option -------> **étudiant(e)** *
scanner chemin **option** ----------> local
supprimer un chemin entre option -------> **local**

Etudiant(e)

lister section
scanner chemin **section** ----------> étudiant(e)*
lister option
supprimer un chemin entre étudiant(e) ----> **option**
scanner chemin **option** ----------> heure
supprimer un chemin entre option ---------> **heure**
scanner chemin **option** ----------> professeur
supprimer un chemin entre option ---------> **professeur**
scanner chemin **option** ----------> local
supprimer un chemin entre option ---------> **local**

Section

lister section
scanner chemin **section** ----------> obligatoire
supprimer un chemin entre section ------> **obligatoire**
scanner **obligatoire** ----------> professeur
supprimer un chemin entre obligatoire --> **professeur**
scanner chemin **obligatoire** ----------> local
supprimer un chemin entre obligatoire --> **local**
supprimer un chemin entre obligatoire --> **heure**

Fonction: MODIFICATION

L' élément à introduire est:

Professeur

lister professeur
scanner chemin **professeur** ----------> obligatoire
supprimer un chemin entre professeur ---> **obligatoire**
supprimer un chemin entre obligatoire --> **heure**
scanner chemin **obligatoire** ----------> section
supprimer un chemin entre obligatoire --> **section**
scanner chemin **obligatoire** ----------> local
supprimer un chemin entre obligatoire --> **local**

ou

lister professeur
scanner chemin **professeur** ----------> option
supprimer un chemin entre professeur ---> **option**
supprimer un chemin entre option -------> **heure**
scanner chemin **option** ----------> étudiant(e)
supprimer un chemin entre option -------> **étudiant(e)**
scanner chemin **option** ----------> local
supprimer un chemin entre option -------> **local**

et

lister professeur
lister obligatoire
créer un chemin entre **professeur** ---> **obligatoire**
lister section
créer un chemin entre obligatoire --> **section**
lister local
créer un chemin entre obligatoire --> **local**
créer un chemin entre obligatoire --> **heure**

ou

lister professeur
lister option
créer un chemin entre **professeur** ---> **option**
lister section
scanner chemin **section** ----------> étudiant(e) *
créer un chemin entre option -------> **étudiant(e)**
lister local
créer un chemin entre option -------> **local**
créer un chemin entre option -------> **heure**

Etudiant(e)

lister section
scanner chemin **section** ----------> étudiant(e) *
supprimer un chemin entre **étudiant(e)** -----> **option**
supprimer un chemin entre option ----------> **heure**
scanner chemin **option** ----------> professeur
supprimer un chemin entre option ----------> **professeur**
scanner chemin **option** ----------> local
supprimer un chemin entre option ----------> **local**

et

lister section
scanner chemin **section** ----------> étudiant(e) *
créer chemin entre **étudiant(e)** -----> **option**
lister professeur
créer chemin entre option ----------> **professeur**
lister local
créer chemin entre option ----------> **local**
créer chemin entre option ----------> **heure**

Section

lister section
scanner chemin **section** ----------> obligatoire
supprimer un chemin entre section ------> **obligatoire**
scanner chemin **obligatoire** ----------> professeur
supprimer un chemin entre obligatoire --> **professeur**
scanner chemin **obligatoire** ----------> local
supprimer un chemin entre obligatoire --> **local**
supprimer un chemin entre obligatoire --> **heure**

et

lister section
lister obligatoire
créer un chemin entre **section** ------> **obligatoire**
lister professeur
créer un chemin entre obligatoire --> **professeur**
lister local
créer un chemin entre obligatoire --> **local**
créer un chemin entre obligatoire --> **heure**

4.1.2.Fonctions de Recherche dans la B.D.

Nous écrivons en caractères gras les arguments d' interface à introduire par l' utilisateur.

Fonction: EDITER

L' élément à introduire est:

Professeur
lister professeur

Section
lister section

Etudiant(e)
lister section
scanner chemin **section** --------> étudiant(e) *

Obligatoire
lister commun

Option
lister option

Local
lister local

Fonction: LISTE_VIDE

L' élément à introduire est:

Professeur
lister professeur
scanner chemin **professeur** -----> obligatoire *
obligatoire ----> heure *

professeur -----> option *
option ---------> heure *

Section
lister section
scanner chemin **section** --------> obligatoire *
obligatoire ----> heure *

Etudiant(e)
lister section
scanner chemin **section** --------> **étudiant(e)** *
étudiant(e) ----> option *
option ---------> heure *

Local

lister local

scanner chemin **local** ----------> obligatoire *

obligatoire ----> heure *

local ----------> option *

option ---------> heure *

Fonction: CONSULTATION

L' élément à introduire est:

Professeur

lister professeur

scanner chemin **professeur** -----> obligatoire *

obligatoire ----> section *

obligatoire ----> local *

obligatoire ----> heure *

professeur -----> option *

option ---------> étudiant(e) *

option ---------> local *

option ---------> heure *

Section

lister section

scanner chemin **section** --------> obligatoire *

obligatoire ----> professeur *

obligatoire ----> local *

obligatoire ----> heure *

Etudiant(e)

lister section

scanner chemin **section** --------> **étudiant(e)** *

étudiant(e) ----> option *

option ---------> professeur *

option ---------> local *

option ---------> heure *

4.2. Synthèse des primitives de la B.D.

4.2.1.Fonctions de Mise à Jour de la B.D.

Les primitives de mises à jour se divisent en 2 groupes:

CREATION de chemins

professeur ---> obligatoire
professeur ---> option

obligatoire --> section
obligatoire --> professeur
obligatoire --> local
obligatoire --> heure

option -------> étudiant(e)
option -------> professeur
option -------> local
option -------> heure

étudiant(e) --> option

section ------> obligatoire

NB: la création des chemins peut se faire dans un sens ou dans l' autre sans influencer les primitives de consultation. Nous conservons uniquement les primitives en gras.

SUPPRESSION de chemins

professeur ---> obligatoire
professeur ---> option

obligatoire --> section
obligatoire --> professeur
obligatoire --> heure
obligatoire --> local

option -------> étudiant(e)
option -------> professeur
option -------> heure
option -------> local

étudiant(e) --> option

section ------> obligatoire

NB: la suppression des chemins peut se faire dans un sens ou dans l' autre sans influencer les primitives de consultation. Nous conservons uniquement les primitives en gras.

4.2.2.Fonctions de Recherche dans la B.D.

Les primitives de consultation se répartissent en 2 groupes:

SCANNER les chemins

professeur -----> obligatoire *
professeur -----> option *

section --------> obligatoire *
section --------> étudiant(e) *

étudiant(e) ----> option *

obligatoire ----> professeur *
obligatoire ----> heure *
obligatoire ----> section *
obligatoire ----> local *

option ---------> professeur *
option ---------> heure *
option ---------> étudiant(e) *
option ---------> local *

local ----------> obligatoire *
local ----------> option *

LISTER des entités

professeur
section
étudiant(e)
obligatoire
option
local

4.3.Contraintes.

Après avoir défini toutes les primitives d' accès, nous allons définir des primitives de contrôle de conformité aux contraintes.

Ces contrôles sont de deux types:

- post-contrôle: détermine si ce qu' on a entré est cohérent avec les contraintes définies dans l' analyse fonctionnelle,

- contrôle de fin: vérifie si l' horaire d' un élément est ou n' est pas complètement défini.

1.Post-Contrôle.

CT1:

R1: scanner **professeur** ----------> obligatoire
R2: scanner **heure** ----------> obligatoire
R3: scanner **section** ----------> obligatoire

ou

R1: scanner **professeur** ----------> option
R2: scanner **heure** ----------> option
R3: scanner **section** ----------> étudiant(e)
scanner étudiant(e) ----------> option

Vérifier que l' intersection des trois règles est vide.

CT2:

R1: scanner **professeur** ----------> obligatoire
R2: scanner **heure** ----------> obligatoire
R3: scanner **section** ----------> obligatoire
R4: scanner **local** ----------> obligatoire

Vérifier que l' intersection des quatre règles est vide.

et

R1: scanner **professeur** ----------> option
R2: scanner **heure** ----------> option
R3: scanner **section** ----------> étudiant(e)
scanner **étudiant(e)** ----------> option
R4: scanner **local** ----------> option

Vérifier que l' intersection des quatre règles est vide.

CT3:

Cette contrainte n' a pas besoin d' être explicitée; elle est implicite.

En effet, on définit dans la B.D. un chemin 1-N entre section et étudiant(e).

CT4:

R1: scanner **section** ----------> étudiant(e)
scanner **étudiant(e)** ----------> option
R2: scanner **heure** ----------> option

Vérifier que l' intersection des deux règles est vide.

CT5:

R1: scanner **section** ----------> obligatoire
R2: scanner **heure** ----------> obligatoire

Vérifier que l' intersection des deux règles est vide.

2.Contrôle de Fin.

La fin d' un horaire doit être uniquement SIGNALEE.

Les conditions de fin d' horaire peuvent varier.

Exemples de condition de fin d' horaire:

Professeur

Nous considérons qu' un professeur achève son horaire en donnant 20 heures de cours.

Section + Etudiant(e)

Nous considérons qu' une section et des étudiants ne suivent pas plus de 32 heures / semaine.

4.4.Bibliographie.

[4.1.] J.-L. HAINAUT

NDBS: A Simple Data Base System for Small Computer

Institut d' Informatique des Facultés Universitaires de Namur - 1987

[4.2.] J.-L. HAINAUT

CONCEPTION ASSISTEE DES APPLICATIONS INFORMATIQUES 2. - Conception de la base de données

MASSON Presses Universitaires de Namur - 1986

CRITIQUES de l' ETUDE

5.1.Critique Ergonomique.

Pour vérifier la qualité du logiciel, nous allons comparer aux réponses provenant du développement de notre application les réponses officielles aux questions posées dans l' aide-mémoire publié par l' INRIA [5.1.].

5.1.1.Pré-Requis.

Caractéristiques des utilisateurs.

Les utilisateurs potentiels sont des secrétaires ou des professeurs de l' enseignement secondaire belge.

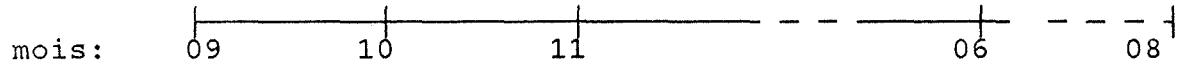
Ces responsables des horaires étaient dans le passé confrontés à un trop gros problème; l' outil informatique essaie de réduire l' énormité de la vision du problème.

Niveau d' expérience des utilisateurs.

Nous faisons l' hypothèse qu' il s' agit d' utilisateurs inexpérimentés.

En effet, on se trouve en face d' utilisateurs mixtes, c' est-à-dire qu' au début de la réalisation de l' horaire, les utilisateurs sont inexpérimentés mais que, tout au long du développement de cet horaire, ils acquièrent de l' expérience.

Le cycle peut se représenter comme suit:



- début 09: utilisateurs inexpérimentés,
- moitié 09 à fin 10: utilisateurs expérimentés,
- 11 à 08: perte de l' expérience.

Analyse des tâches.

Les objets manipulés sont les entités:
- professeur,
- étudiant(e),
- section,
- option,
- obligatoire.

Les opérations appliquées à ces objets sont:
- création de chemins entre deux entités,
- suppression de chemins entre deux entités,
- modification de chemins entre deux entités,
- édition de toutes les valeurs d' entités,
- recherche à travers des chemins de valeurs d' entités.

Séquencement.

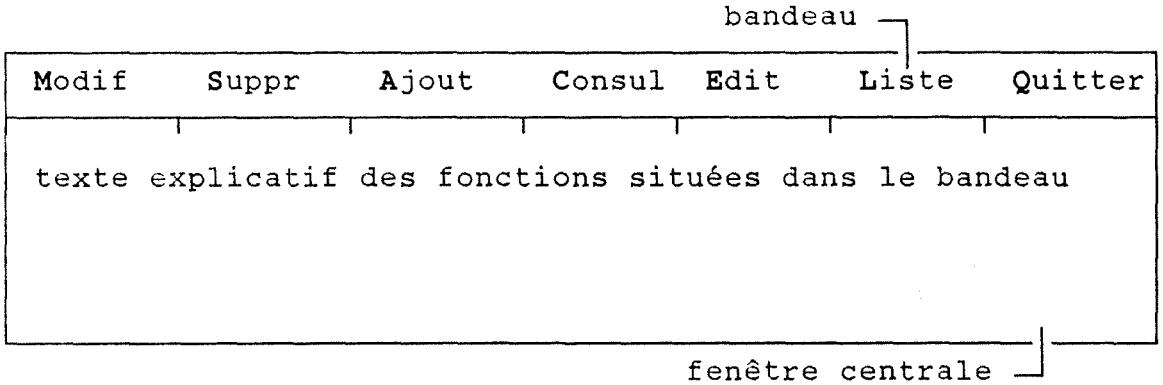
Le séquencement des différents écrans est rigoureux et se fait toujours dans le même ordre pour les mêmes hypothèses.

Ce séquencement suit la logique du raisonnement humain nous a-t-il semblé et permet à l' utilisateur habitué de ne plus lire le titre de l' écran pour pouvoir effectuer son choix.

Exemple de séquencement:

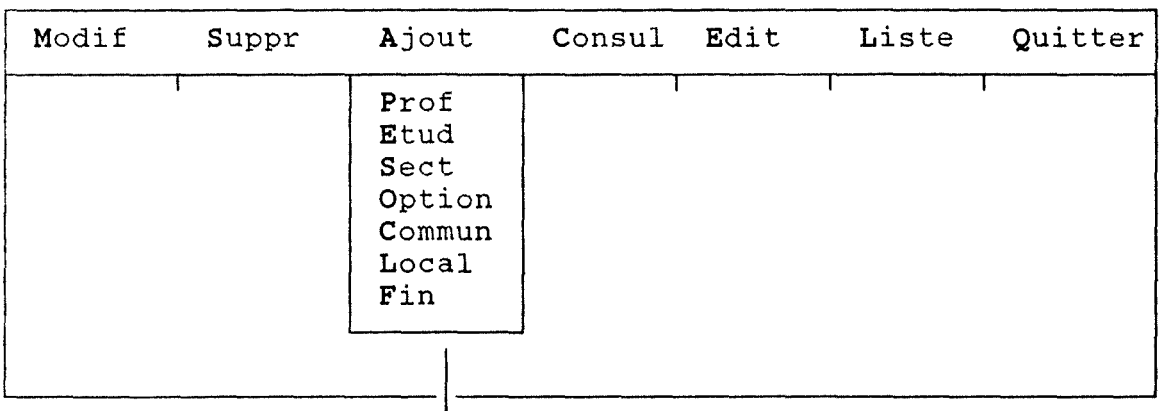
La première opération consiste à choisir l' action souhaitée .

Ce choix se fait parmi les fonctions proposée dans le bandeau dont la description se trouve dans la fenêtre centrale.



Une fois le choix de la fonction fait, une petite fenêtre apparaît sous la fonction sélectionnée.

Prenons l' exemple d' un Ajout:

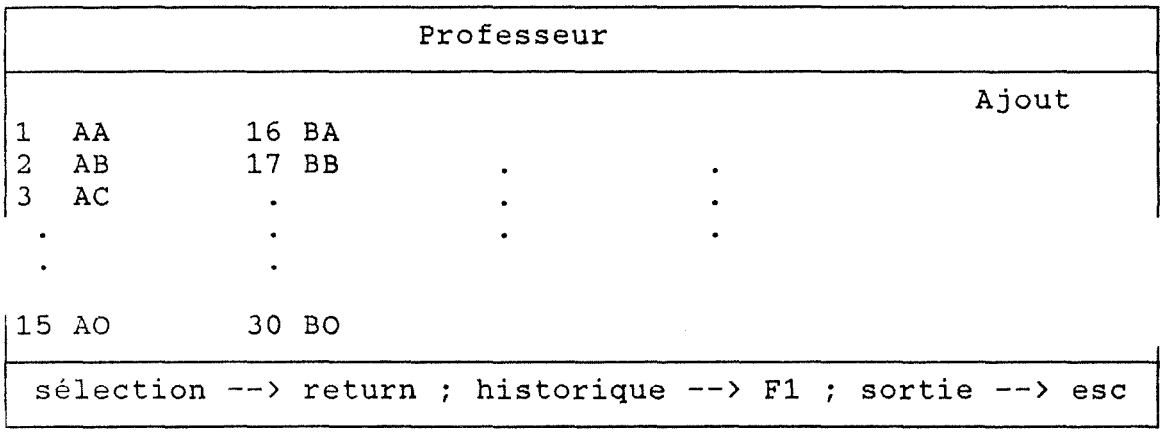


fenêtre déroulante apparaissant lors de la sélection de Ajout

L' apparition de cette fenêtre correspond à la prise de décision de la façon d' aborder le problème.

Prenons le cas où le premier élément est un professeur.

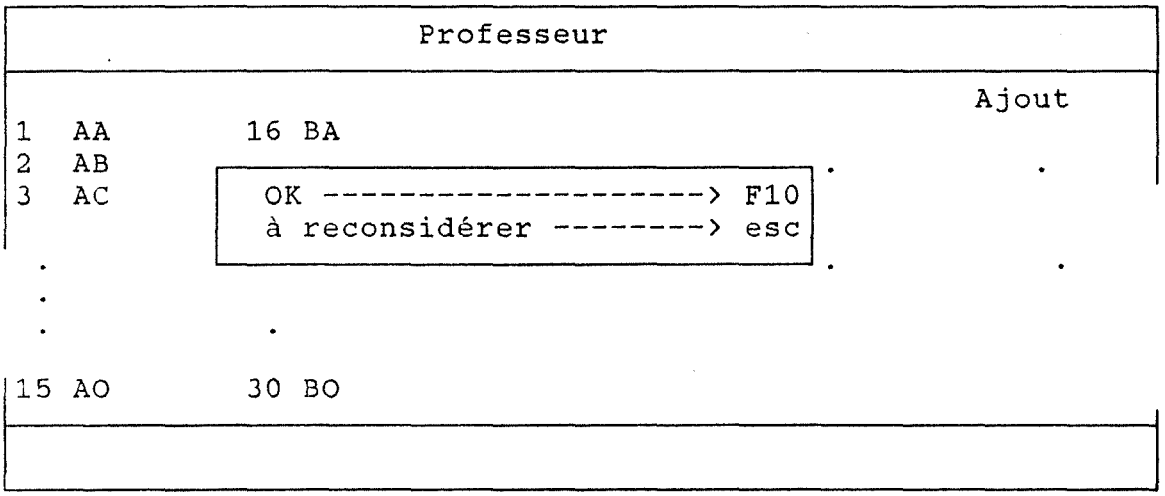
Dans ce cas l' écran professeur apparaît:



L' utilisateur va sélectionner le numéro et le nom du professeur qui sera l' origine d' un nouvel élément d' horaire.

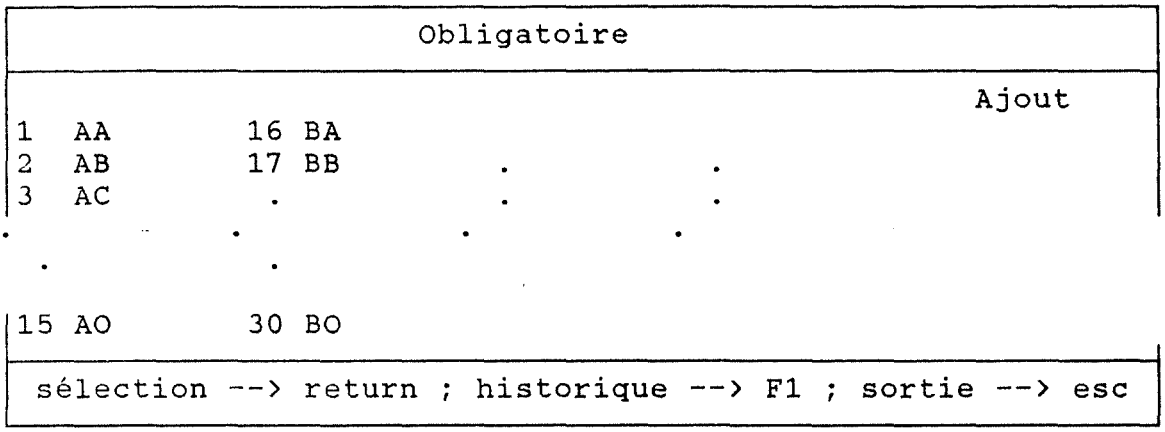
Le déplacement se fait à l' aide des flèches et la sélection à l' aide de la touche "return".

A ce moment apparaît l' écran de confirmation ou de rejet de la sélection du professeur:



Si on reconsidère, on retombe sur l' écran professeur.

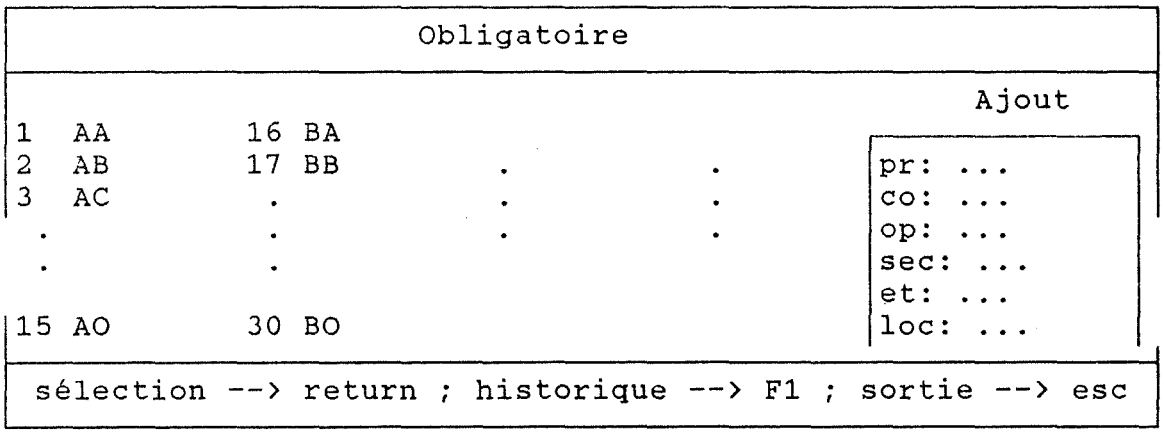
Si c' est OK, apparaît l' écran obligatoire:



Si l' utilisateur a dû s' absenter entre cet écran et le précédent, il peut faire appel à la touche "historique" (F1).

Dans la suite, nous ne reprendrons plus cette commande quoique toujours utilisable avant toute sélection.

Voici l' écran après sélection de la touche F1:



Si nous reprenons en détails cette mini fenêtre, nous avons:

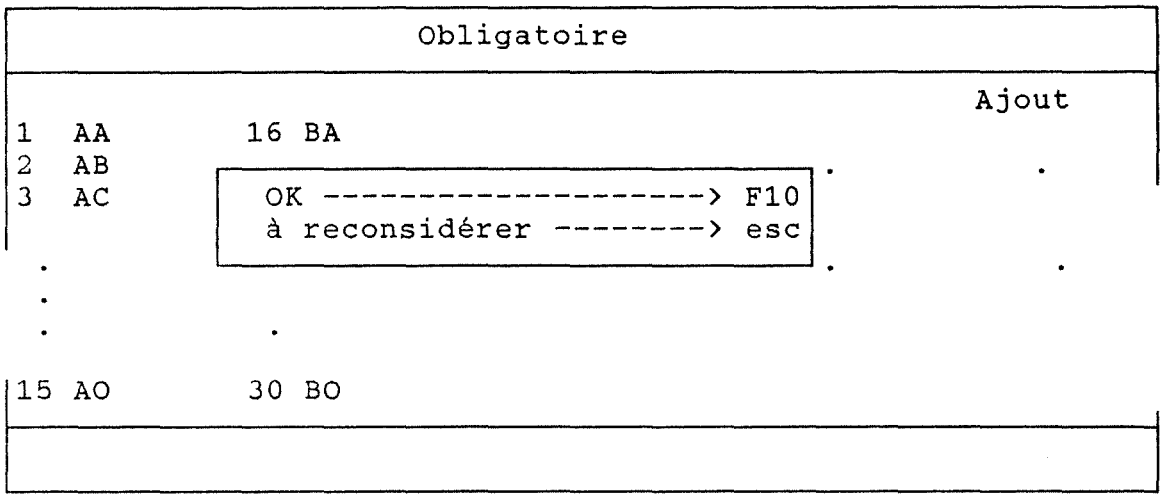
pr: < numéro du professeur > < abrégé du nom >,
co: < numéro du cours commun > < abrégé de l' intitulé >,
op: < numéro du cours à option > < abrégé de l' intitulé >,
sec: < numéro de la section > < nombre d' étudiants >,
et: < numero d' étudiant > < abrégé du nom >,
loc: < numéro du local > < contenance maximum du local >.

Nous pouvons plus facilement effectuer notre choix.

L' utilisateur va sélectionner le numéro et le nom du cours obligatoire.

Le déplacement se fait à l' aide des flèches et la sélection à l' aide de la touche "return".

A ce moment apparaît à l' écran la fenêtre de confirmation ou de rejet de la sélection du cours obligatoire.

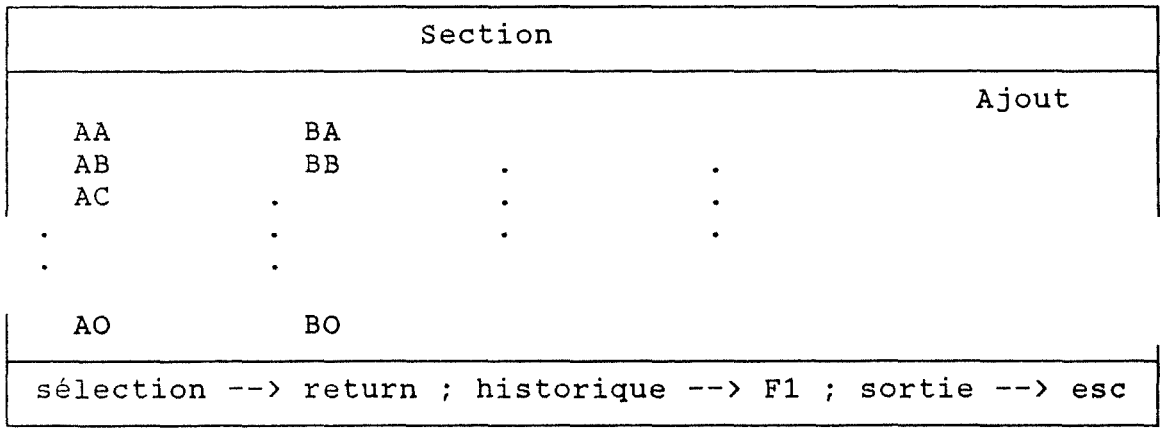


Si l' utilisateur n' a saisi aucune donnée sur l' écran obligatoire (touche "esc"), alors l' écran option apparaît.

Dans le cas contraire, on passe directement à l' écran section.

Si on reconsidère, on retombe sur l' écran obligatoire.

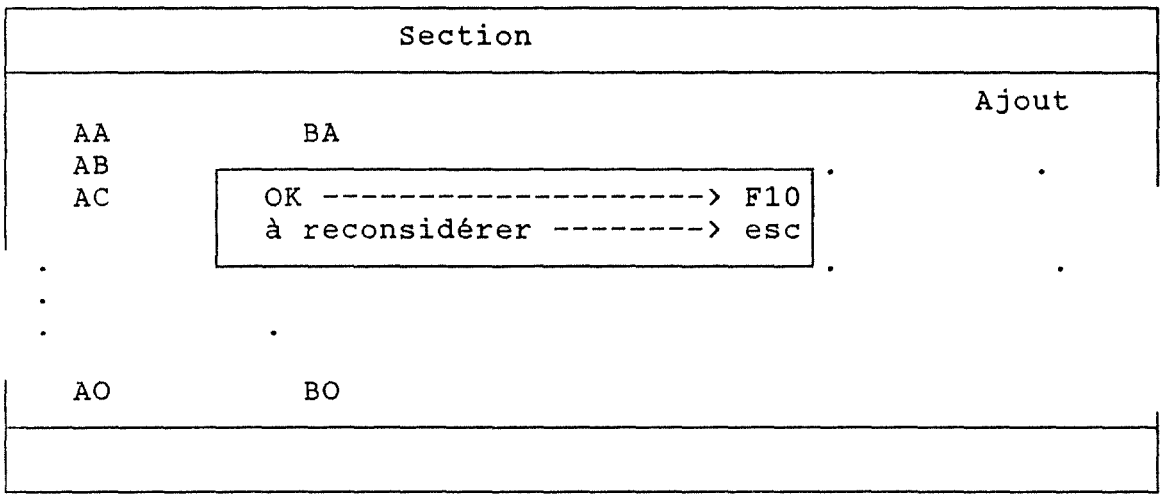
Si c' est OK, apparaît l' écran section:



L' utilisateur va sélectionner le numéro de la section.

Le déplacement se fait à l' aide des flèches et la sélection à l' aide de la touche "return".

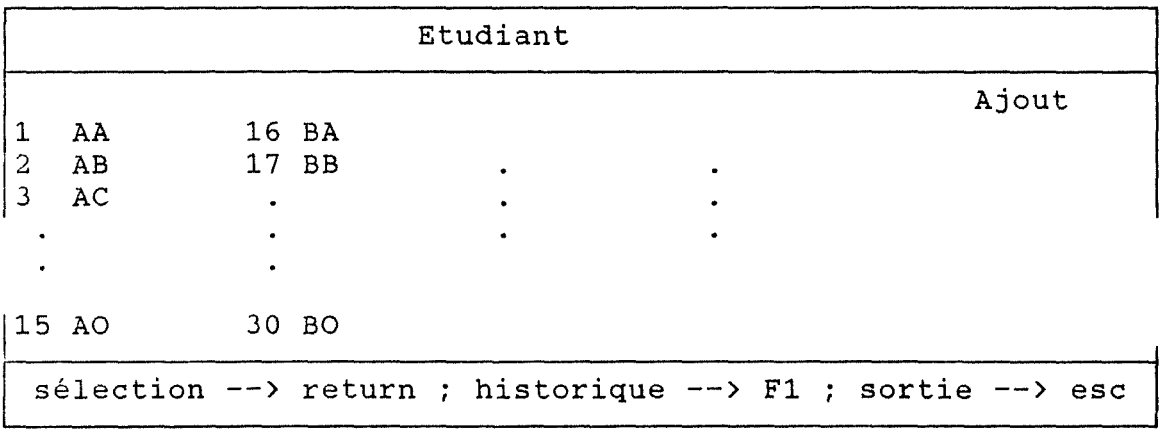
A ce moment apparaît l' écran de confirmation ou de rejet de la sélection de la section.



Si on reconsidère, on retombe sur l' écran section.

Si c' est OK, apparaît l' écran étudiant ou l' écran local suivant qu' on a eu affaire respectivement à un cours à option ou à un cours obligatoire:

1.Etudiant

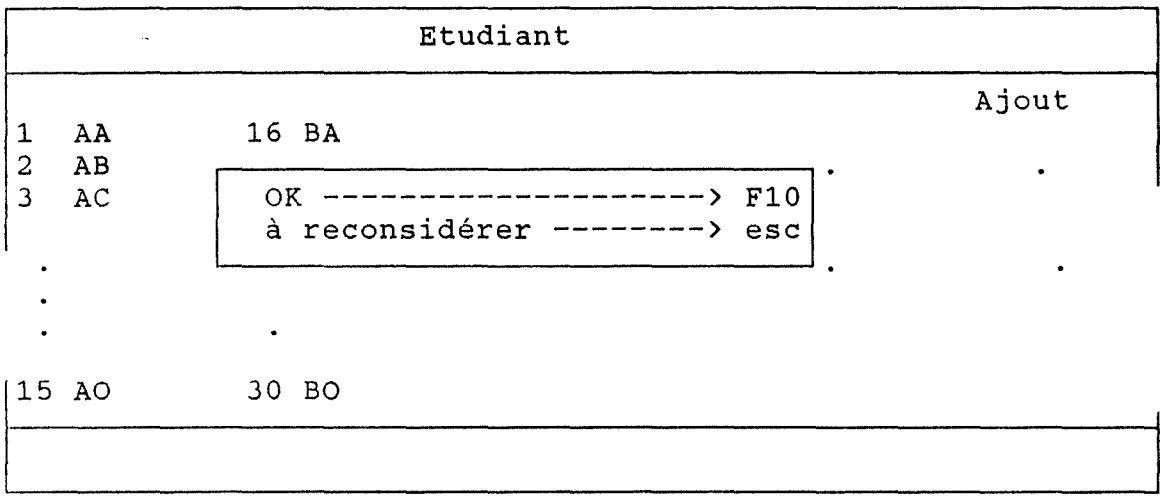


L' écran étudiant affiche les renseignements concernant la section préalablement définie grâce à l' écran section.

L' utilisateur va sélectionner le numéro de l' étudiant.

Le déplacement se fait à l' aide des flèches et la sélection à l' aide de la touche "return".

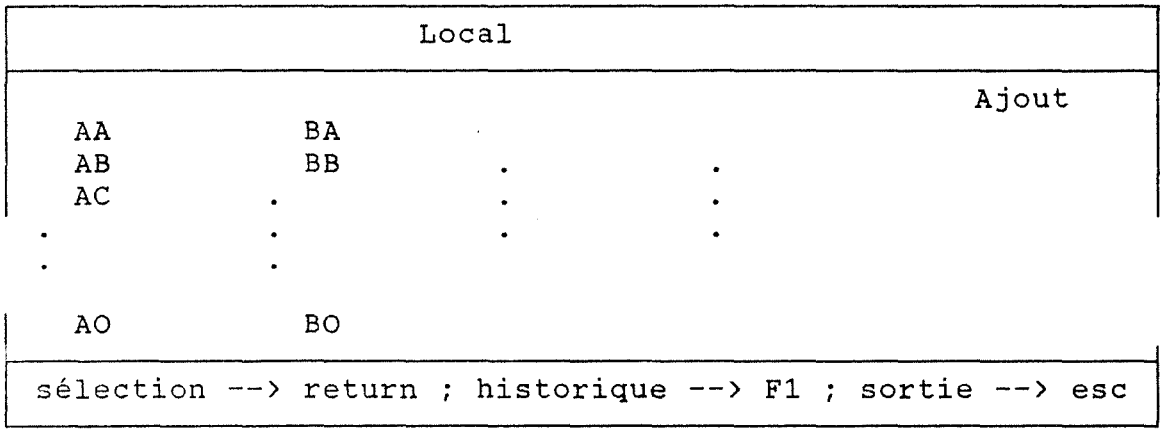
A ce moment apparaît l' écran de confirmation ou de rejet de la sélection de l' étudiant.



Si on reconsidère, on retombe sur l' écran étudiant.

Si c' est OK, apparaît l' écran local.

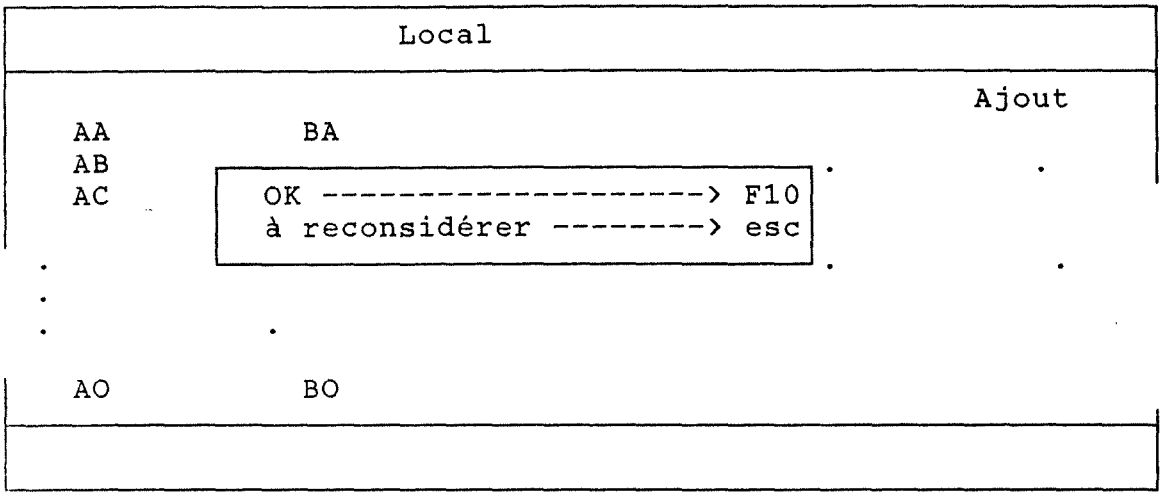
2. Local



L' utilisateur va sélectionner le numéro du local.

Le déplacement se fait à l' aide des flèches et la sélection à l' aide de la touche "return".

A ce moment apparaît l' écran de confirmation ou de rejet de la sélection du local.



Si on reconsidère, on retombe sur l' écran local.

Si c' est OK, apparaît la grille horaire avec les indisponibilités de professeur et de locaux:

	1	2	3	4	5	6	7	8
lun								
mar								
mer								
jeu								
ven								

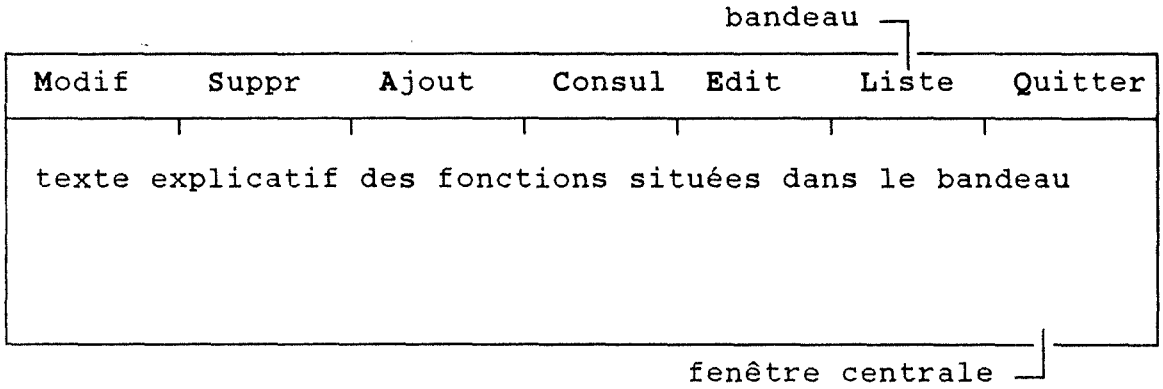
légende:

= heure indisponible pour le professeur

= local indisponible

A ce moment, l' utilisateur déplace le curseur à l' aide des flèches et se place dans la case choisie.

Une fois cet élément d' horaire terminé, nous retombons sur l' écran initial:



5.1.2.Critères de Conception.

Compatibilité entre solution manuelle et informatique.

Cette compatibilité porte sur différents points:

- écrans / papiers:

Deux écrans de saisies de données pour traitement existent:

* affichage de toutes les valeurs d' entités,
* affichage d' une grille horaire.

Les listes sont des documents qui sont habituellement utilisés dans les domaines du bureau.

Nous avons veillé à afficher sur un seul écran toutes les données nécessaires à l' utilisateur.

En ce qui concerne la grille horaire, cela ressemble à la meilleure modélisation des données faite à ce jour.

- vocabulaire:

Nous avons veillé à utiliser toujours le vocabulaire le plus vulgaire possible pour ne pas décourager les non-initiés alors que les habitués ne prennent pas la peine de lire les écrans attentivement.

En ce qui concerne les données à saisir, nous les représentons par un numéro identifiant difficile à retenir (numéro de l' étudiant, numéro du professeur, numéro du cours à option, numéro du cours obligatoire) ou non (numéro de la section, numéro du local) accompagné dans le premier cas d' un nom non identifiant mais représentatif de l' entité pour l' utilisateur.

Homogénéité entre les différentes "informations".

Tous les écrans de saisies de données autres que les données "heure" s' affichent sur les mêmes écrans, de la même façon, avec les mêmes possibilités de déplacement; seul les titres de ces écrans de saisies sont changés.

La touche "return" correspond à une sélection alors que la touche "F10" correspond à une validation de la donnée saisie et la touche "escape" correspond au rejet.

Cette touche "F10" n' existe que pour les saisies des éléments d' horaire; en effet, c' est là qu' on peut craindre le plus d' erreurs de frappe ou de lassitude.

Après affichage d' un texte demandant confirmation ou recommencement, un nouveau choix de la part de l' utilisateur est possible tant que cette touche "F10", n' a pas été sélectionnée (dans le contexte présenté précédemment).

Niveaux d' expérience.

La possibilité d' un choix de niveau d' expérience n' a pas été envisagée.

En effet, outre le fait qu' il n' est pas certain que l' utilisateur inexpérimenté se transforme en utilisateur expérimenté, nous considérons que le temps de sélection dans un menu n' est rien en comparaison du temps de traitement dans la B.D.

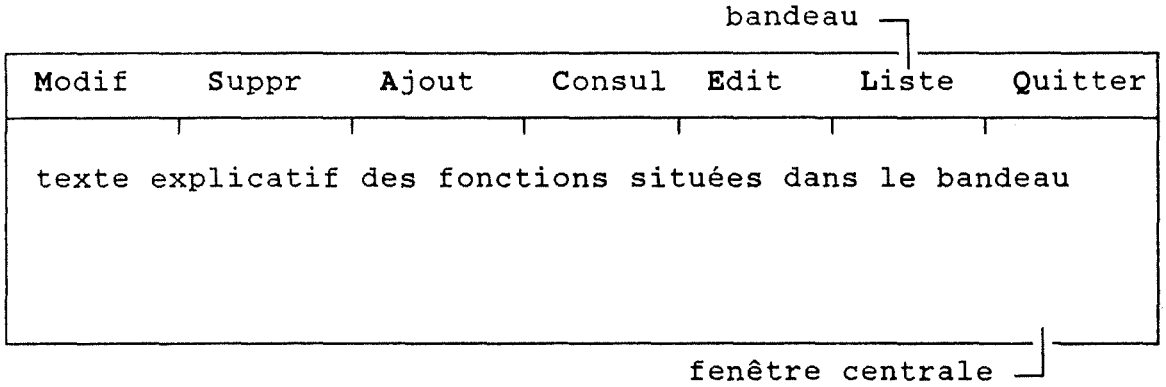
En effet l' écran de menu est unique et travaille en mémoire d' écran.

De plus, les écrans de saisie travaillent en mémoire d' écran et donc à affichage rapide alors que les données sorties de la B.D. prennent beaucoup de temps, surtout si on travaille avec des disques souples.

Guidage.

L' utilisateur bénéficie d' un guidage uniquement au niveau de l' écran initial.

Ce guidage se situe dans la fenêtre centrale:



Nous avons prévu les rudiments d' un guidage fonctionnel mais pas le guidage d' utilisation: nous considérons les opérations possibles élémentaires et logiques pour l' utilisateur qui, à défaut de connaître l' informatique, connaît très bien la gestion des horaires.

Minimum d' actions.

Nous avons veillé, grâce à la base de données, à ce que les données entrées ne doivent plus être introduites à nouveau.

Il y a réutilisation des données entrées.

Exemple:

Prenons le cas de la création d' un élément d' horaire.

Le premier élément est le professeur.

Nous sélectionnons le professeur.

Ce professeur est mémorisé une fois pour toute, il est d' ailleurs possible de l' afficher dans une petite fenêtre d' aide.

Après le professeur, c' est au tour du cours obligatoire ou cours à option d' être sélectionné.

Il y a création d' un chemin entre ces deux valeurs de T.E.

Ensuite, il y a sélection de la section.

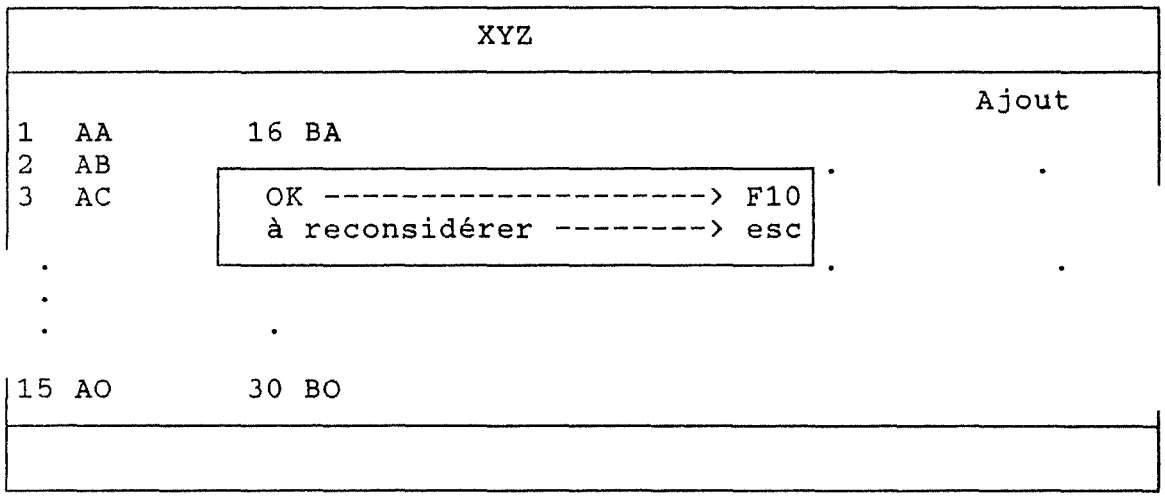
Il y a création d' un chemin entre section et cours obligatoire.

Implicitement, il y a communication entre la section et le professeur.

Il en va de même pour toutes les entités.

Nous veillons à déceler les erreurs le plus tôt possible pour perdre le moins de temps possible.

C' est pour cette raison que après chaque saisie, nous avons une étape de confirmation ou rejet.



Nous évitons d' utiliser des compositions de touches pour une action, quoique cela pourrait avoir un intérêt dans le cas d' actions trop souvent répétées et qui demandent, malgré tout, de la concentration (dans le cas de suppression par exemple).

Traitement des erreurs.

Les erreurs sont traitées après tout contact avec l' utilisateur et donc après chaque écran, par l' affichage d' un écran signalant une erreur.

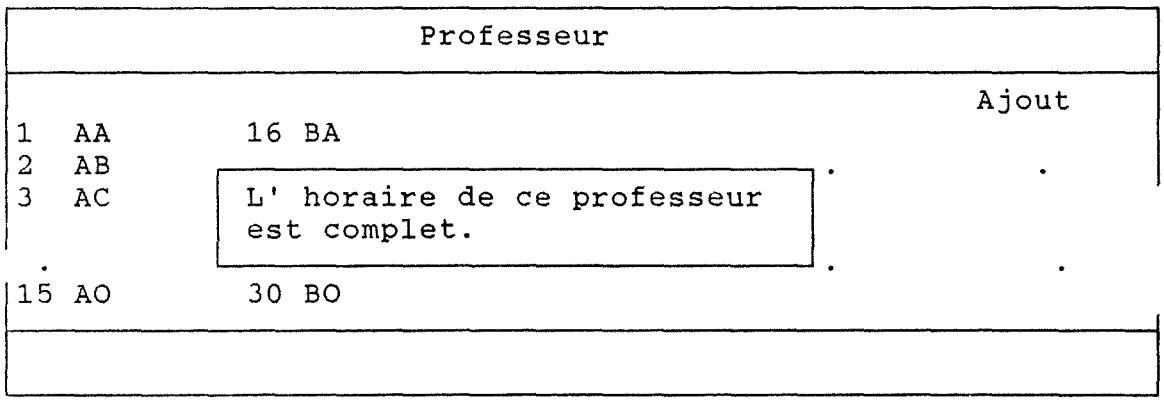
Exemple:

Jusqu' à présent, nous avons considéré que tout se passait bien.

Mais des problèmes peuvent arriver lors de la création d' un chemin déjà existant, lors de la suppression d' un chemin inexistant, ...

Dans ce cas, une fenêtre en surimpression apparaît où on énonce l' erreur de traitement correspondante (la liste de ces messages est énoncée en 3.3.2.).

Par exemple, dans le cas d' un professeur qui a déjà toutes ses heures de cours placées:



Les erreurs possibles sont d' une part les erreurs de frappe sur la saisie à faire; en effet, nous avons fait en sorte que ne puissent être saisies que des données permettant un travail cohérent de la B.D. (bien que n' interdisant pas des incohérences par infraction aux contraintes);et d' autre part des erreurs d' intentions qui viennent d' une mauvaise compréhension des commandes, dans ce cas, nous n' avons rien prévu.

Après chaque écran, il y a un prise de décision: "la donnée saisie est-elle la bonne ? OUI ou NON".

Cette question est un passage obligé pour continuer le traitement.

5.1.3.Menus.

Nous utilisons exclusivement ce type de dialogue vu l' hypothèse que nous avons faite quant au type d' utilisateurs.

Sélection unique par pointage direct.

Nous offrons la possibilité à l' utilisateur de sélectionner un article dans un menu à l' aide des flèches qui permettent un choix plus précis et donc plus rapide ou (pour les habitués) par le premier caractère en surbrillance de l' option choisie.

Un seul choix est possible sur chaque écran.

Toutes les options ont été prévues et se trouvent soit dans le bandeau soit dans le déroulement.

Codes option.

Le vocabulaire des options (choix) est facilement compréhensible, il est exprimé en français.

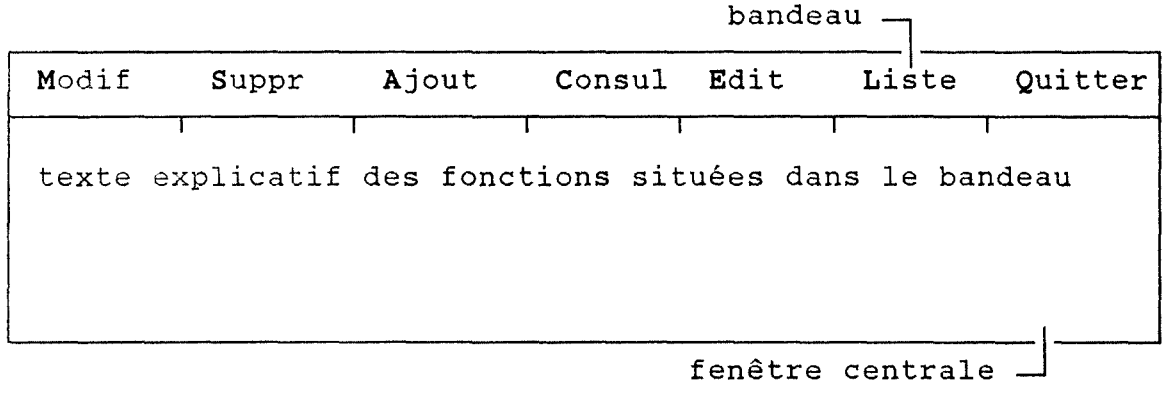
Pour l' utilisateur expérimenté, le choix d' une option se fait par le code option qui est la première lettre du mot choisi et qui est en surbrillance.

Ordre des options.

L' ordre de rangement des options principales, dans le bandeau est:

- mise à jour de chemins entre les T.E.,

- recherche d' information.



L' ordre dans lequel on a placé les sous-options est un des ordres qui placent en tête la sous-option la plus utilisée.

Modif	Suppr	Ajout	Consul	Edit	Liste	Quitter
Prof Etud Sect Option Commun Local Fin	Prof Etud Sect Option Commun Local Fin	Prof Etud Sect Option Commun Local Fin	Prof Etud Sect Option Commun Local Fin	Prof Etud Sect Option Commun Local Fin	Prof Etud Sect Local Fin	

fenêtre déroulante apparaissant lors de la sélection de modification au niveau 1

Bien que les flèches aillent dans les deux sens, l' utilisateur a tendance a n' aller, au début, que dans un sens puisque ce sont principalement les inexpérimentés qui utilisent des flèches.

Titre des menus.

Chaque écran comporte un titre qui permet à l' utilisateur de comprendre où il se situe dans l' évolution de l' application.

Feed-back.

Le passage d' un écran à un autre est estimé suffisamment rapide.

Pour en arriver là, on a utilisé les affichages à partir de la mémoire d' écran.

La recherche de données dans la B.D. prend beaucoup plus de temps et particulièrement dans le cas où on travaille avec des disques souples.

Nous n' avons pas prévu de messages informant d' une attente prolongée.

C' eût été éventuellement intéressant pour l' attente lors de l' impression sur imprimante.

5.1.4.Entrées.

Action minimale.

Les utilisateurs ne doivent JAMAIS entrer des valeurs au clavier; seules des sélections de valeurs imprimées sur écran sont à faire.

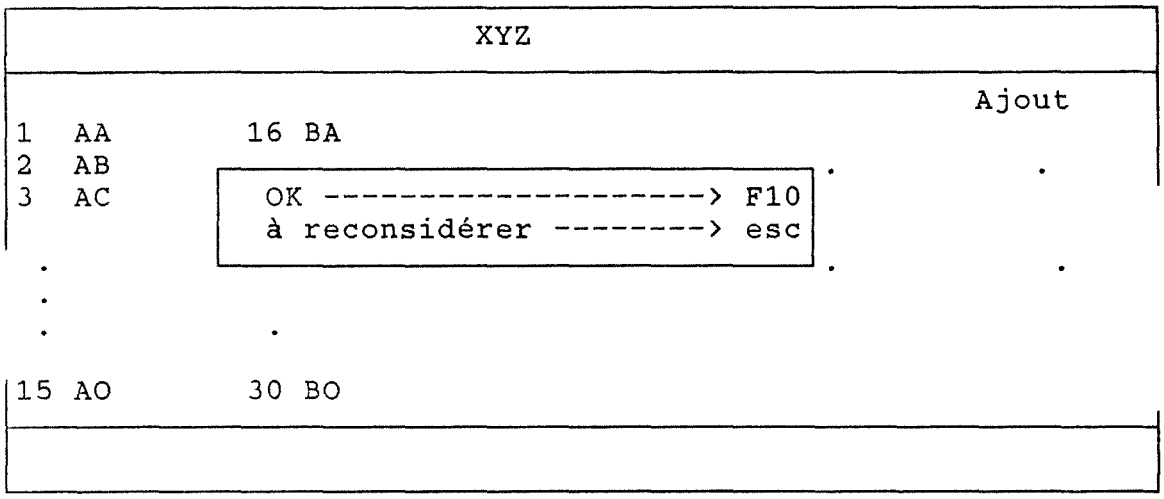
Les utilisateurs ne doivent JAMAIS sélectionner plusieurs fois les mêmes valeurs pour une même application.

Cette approche minimise:

- la mémorisation,
- le risque d' erreur de saisie.

Confirmation.

Après une sélection par l' utilisateur, apparaît un fenêtre demandant confirmation ou non.



La sélection de données se fait par la touche "return" alors que la confirmation se fait par la touche "F10" et la rejet s' enclenche avec la touche "escape".

Achèvement de l' entrée.

Il n' y a qu' une seule saisie par écran.

A la fin de chaque écran, donc de chaque saisie, il y a possibilité de confirmation ou de rejet.

A la fin de chaque application, on revient au menu déroulant principal.

Par application, on pense au traitement d' une partie d' horaire entre toutes les entités mais pour une seule valeur.

Exemple:

Dans le cas de la création d' un élément d' horaire, on crée des chemins entre le professeur et le cours obligatoire ou le cours à option, entre la section et le cours obligatoire ou entre l' étudiant et le cours à option, entre le local et le cours obligatoire ou le coures à option, entre l' heure et le cours obligatoire ou le cours à option.

C' est cela qu' on appelle une application.

Erreurs.

Lors d' une erreur, le logiciel ne peut que demander à l' utilisateur s' il est d' accord ou non.

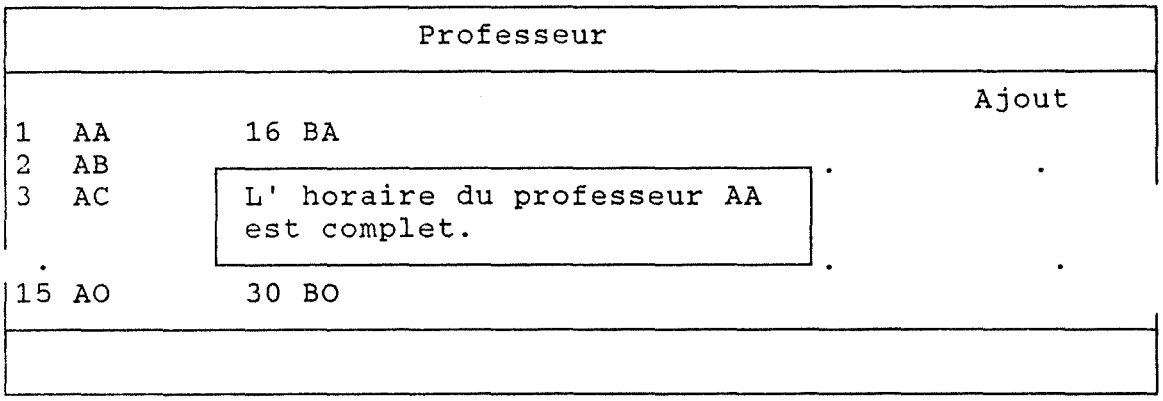
En plus de ces erreurs, il existe les vérifications de contraintes, qui pourraient être assimilées à des erreurs de traitement.

Ces vérifications, lorsqu' elles causent des problèmes, conduisent le plus tôt possible à l' affichage d' un message générique du type d' "erreur" sur l' écran où s' est produit cette erreur.

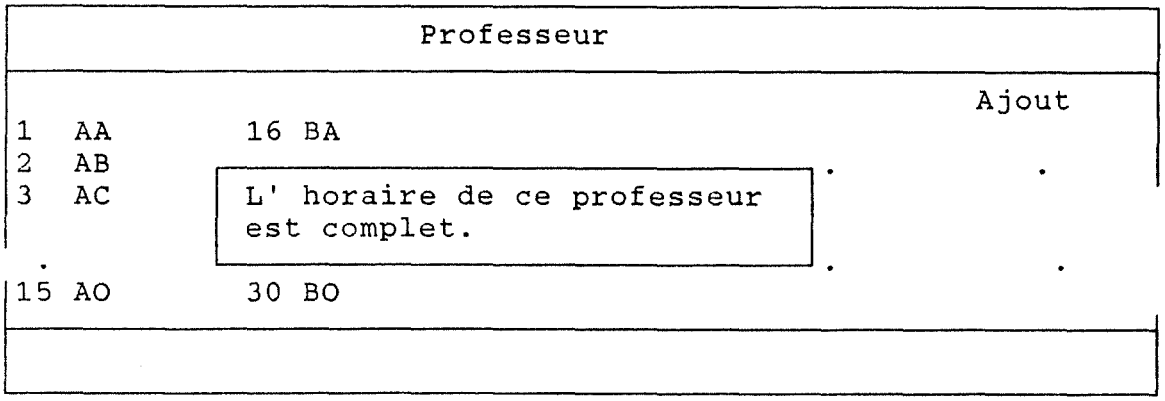
De tels messages ne sont jamais personnalisés aux erreurs qu' ils expliquent, mais sont suffisamment explicites pour permettre une correction.

Exemple:

Dans le cas d' un professeur dont l' horaire est complet, le message qui apparaît n' est pas:



mais bien:



Nous avons toujours affaire à des messages génériques.

Curseur.

Dans tous les cas de mise à jour, le curseur est un bandeau (inverse video) qui recouvre l' information sélectionnée ou une information intermédiaire conduisant à celle souhaitée.

Dans tous les cas de recherche, le curseur est inexistant.

A l' affichage de l' écran menu, le curseur se trouve sur la première option du menu principal.

A l' affichage d' écran de saisie de données, le curseur se trouve sur le premier élément de la liste de valeurs.

A l' affichage de la grille horaire, le curseur se trouve sur la case 4ème heure du mercredi pour permettre un déplacement le plus rapide possible.

5.1.5.Séquences de commande.

Conformité des buts.

Les actions sans danger (ex: création) et les actions dangereuses (ex: supprimer) ont le même degré de complexité.

Exemple:

Dans le cas d' une création, on demande confirmation ou rejet des décisions avec la même constance que dans le cas d' une suppression.

Expérience des utilisateurs.

Aucune distinction n' a été faite quant à l' expérience des utilisateurs en ce qui concerne le séquencement des commandes.

En effet, outre l' hypothèse de départ, les commandes apparaissant dans cette application consistent en des prises de données nécessaires et en un traitement de ces données.

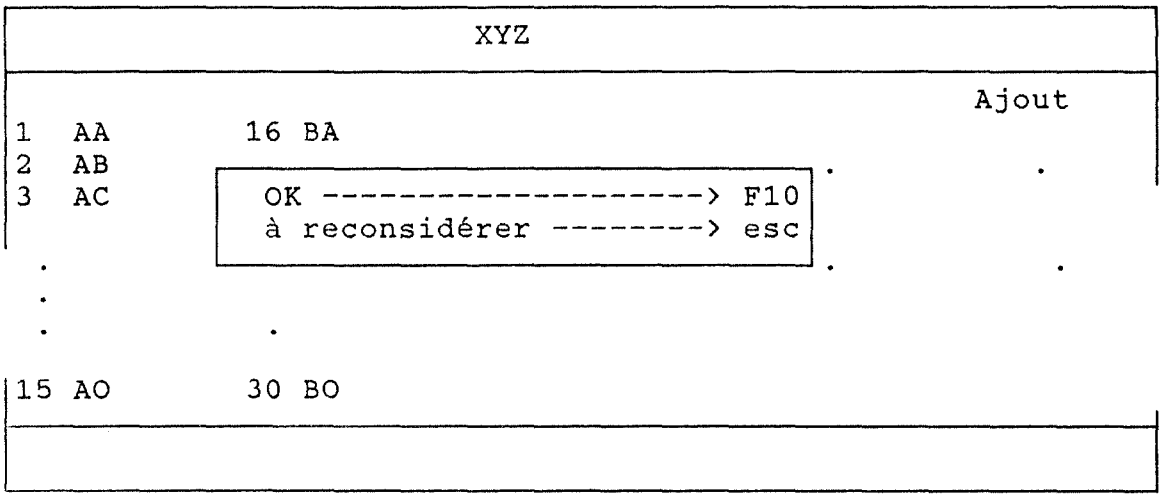
Cohérence et contexte.

Tous les écrans se suivent de la même manière pour des options du déroulement identiques(ex: professeur, section, étudiant, ...) et éventuellement des options du bandeau différentes(ex: ajout, suppression, ...).

Annuler, retour-arrière, reprise, détruire, fin.

On peut annuler c' est-à-dire qu' on peut régénérer un écran sans que la valeur sélectionnée ne soit prise en considération.

En effet, après chaque sélection, un écran demande si le choix est le bon ou non.



Il suffit de reconsidérer son choix.

On n' a pas prévu de retour-arrière.

Si on veut changer une valeur entrée sur un écran précédent, il faut utiliser la fonction modifier.

L' option reprise qui a pour but de revoir toutes les données de l' application et de les modifier si nécessaire n' est pas prévue.

Dans les deux cas précédents, on pense que le risque d' avoir encore un erreur de choix est très faible; la correction est toujours possible mais plus longue.

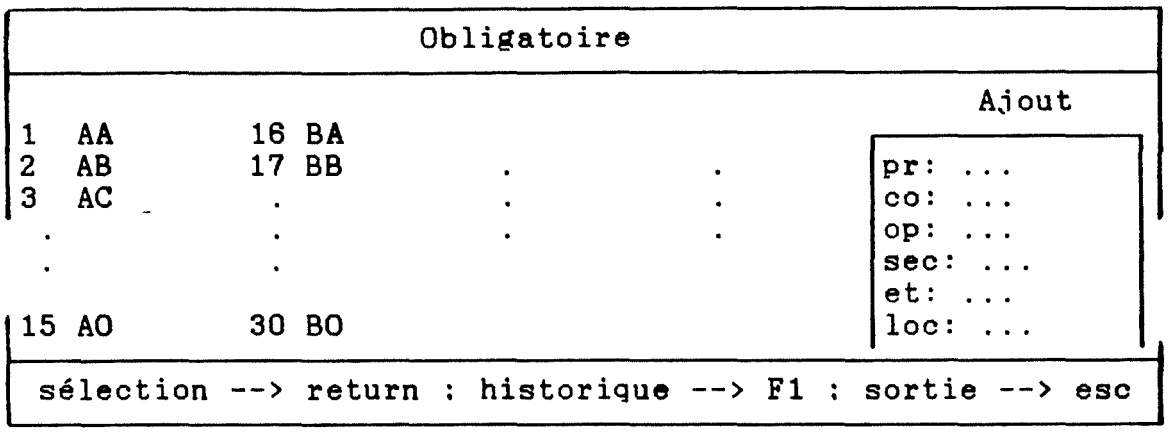
Il nous a semblé qu' il est plus facile de recommencer l' application, qui n' est jamais bien longue, que de poursuivre l' application en cours en y effectuant des modifications.

A la fin de chaque application, si la dernière saisie est confirmée, on retourne au menu principal.

Fonction affichant entrées précédentes.

On a prévu cette fonction pour permettre à l' utilisateur d' interrompre sa tâche en cours.

L' activation de cette fonction se fait par exemple par la touche "F1".



Si nous reprenons en détails cette mini fenêtre, nous avons:

pr: < numéro du professeur > < abrégé du nom >,
co: < numéro du cours commun > < abrégé de l' intitulé >,
op: < numéro du cours à option > < abrégé de l' intitulé >,
sec: < numéro de la section > < nombre d' étudiants >,
et: < numéro d' étudiant > < abrégé du nom >,
loc: < numéro du local > < contenance maximum du local >.
Fin de session.

On termine une session sur le menu principal.

A ce stade, tous les résultats des opérations faites jusqu' à présent sont mémorisés.

5.1.6.Sorties.

Temps de réponse.

Il faut distinguer différents temps de réponses:

- temps de réponse d' affichage: ce temps de réponse descend sous la seconde et est acceptable,
- temps de réponse de traitement: ce temps dépend du temps d' accès au disque, de la façon dont sont placés les articles et du type d' accès aux articles; ce temps est nettement supérieur au temps de réponse d' affichage et dépend de l' importance de l' école.

NB: Nous avons toujours travaillé sur un disque dur.
Nous avons essayé les disquette souples mais cela prenait beaucoup trop de temps.
L' estimation des temps est difficile à faire car ils dépendent du type d' école.
On peut dire, par expérience, que la recherche d' un article dans un fichier de 20 articles(1) n' a jamais dépassé la barre gênante des quatre secondes.

Messages d' attente.

Nous n' avons pas prévu de message d' attente.

Le seul cas où cela aurait pu être utile se présente lors de l' impression.

Cependant, l' application est prévue pour de la mono-utilisation; l' utilisateur, travaillant sur micro-ordinateur, n' est généralement pas éloigné de son imprimante et peut donc surveiller son fonctionnement.

Mais un tel message peut exister.

Corrections des informations à l' écran.

A l' écran sont affichées des valeurs qui sont appelées de la B.D. par des algorithmes prétendument corrects.

Niveau de message.

Nous n' avons pas prévu de messages différents pour des utilisateurs de niveaux différents.

Fonction aide.

Nous aurions pu prévoir une fonction d' aide "F2" qui aurait dépendu de l' endroit où se serait situé l' utilisateur.

Les explications données se seraient adressées aux novices.

Les explications auraient porté suivant les cas:

- sur des problèmes signalés par le logiciel,
- sur des problèmes rencontrés par l' utilisateur.

---

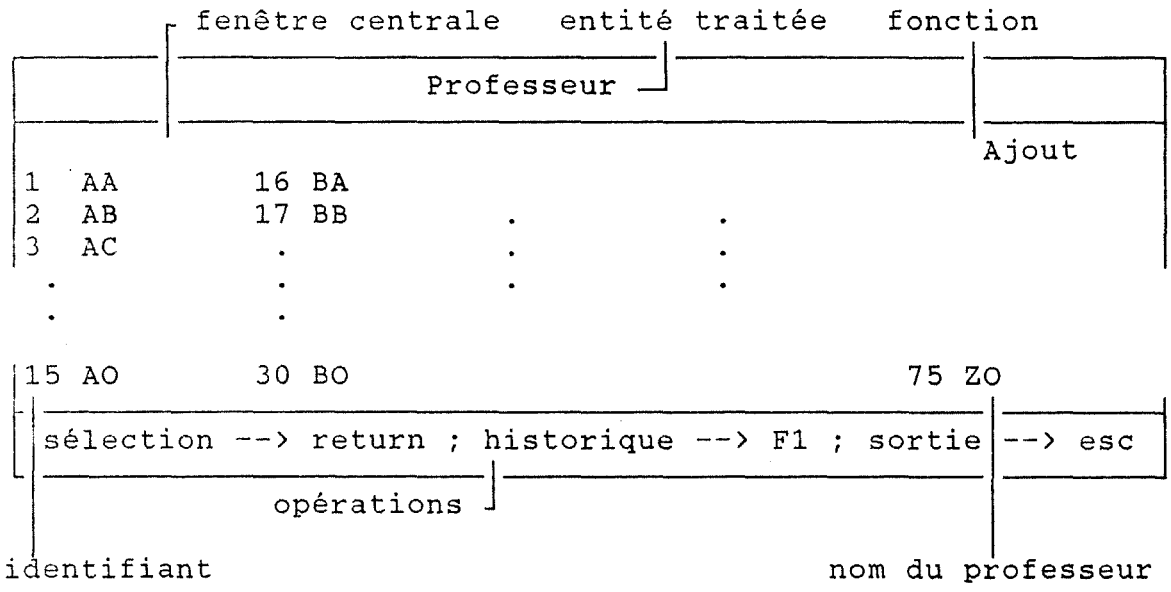
(1) Un fichier dans ce contexte représente toutes les valeurs d' un T.E.
L' écran est organisé pour maximum 75 articles.
Il apparaît que 75 éléments est honnête.
Cela prévoit, en effet, 75 sections, 75 étudiants par section, 75 professeurs, 75 locaux, 75 cours à option, 75 cours obligatoire.

Utilité des informations affichées.

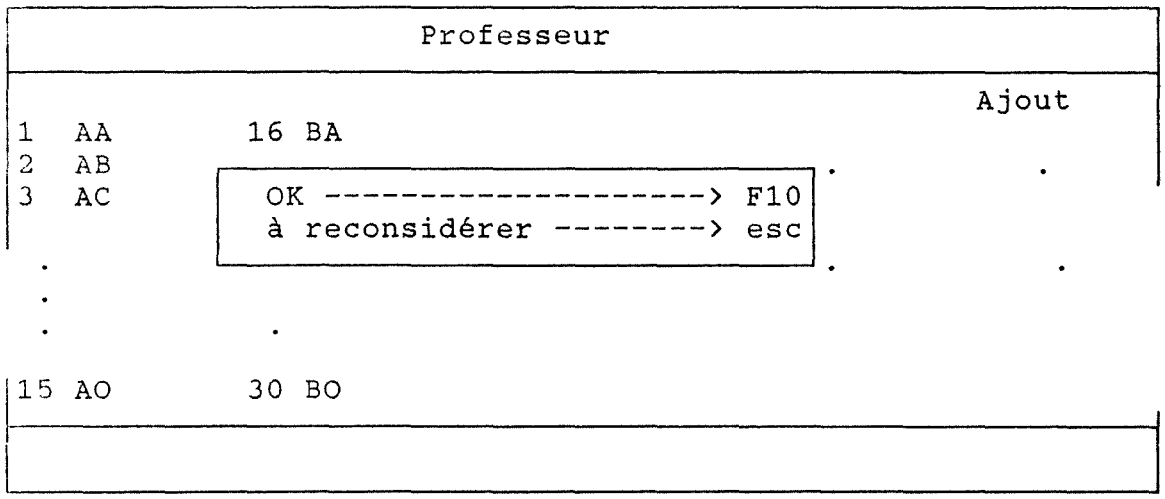
Nous affichons des informations utiles au moment de leur utilisation.

Exemple:

Lors de la sélection d' un professeur, l' utilisateur reçoit l' écran suivant:



Une fois la sélection faite (touche "return"), l' écran se transforme en:



On constate la disparition des opérations qui ne sont plus utilisables à ce moment; les seules admises sont celles apparues dans la petite fenêtre en surimpression.

Dès que leur utilité est passée, elles s' éclipsent.

Formats de sortie.

Les documents de sortie sont ceux qui apparaîtraient lors d' un même travail fait à la main.

Tous les travaux d' un même type ont des sorties d' un même type.

Ils sont donc directement utilisables.

Ecrans facilement compréhensibles.

Il s' agit de voir si l' accès à l' information est aisé.

Toutes nos informations sont présentées sous forme de listes et sur un même écran sans aucun "scrolling".

Les données appartenant à ces listes sont séparées par au moins deux caractères blancs.

```
                        Professeur

                                                          Ajout
1  AAAAAAA  16 BA
2  ABBBBBB  17 BB          .           .
3  ACCCCCC   .             .           .
  .          .             .           .
  .          .

15 AOOOOOO  30 BO

 sélection --> return ; historique --> F1 ; sortie --> esc
```

Regroupement des items.

Comme nous l' avons expliqué en 5.1.2., les items sont groupés suivant leur identifiant.(1)

```
                    Obligatoire

                                                        Ajout
1  math-3
2  math-4
3  math-5
4  math-7
5  math-9
6  latin-4
7  latin-5
8  fran-4
9  fran-5

15 AO        30 BO

 sélection --> return ; historique --> F1 ; sortie --> esc
```

Malheureusement, on constate qu' il n' y a pas de séparation entre les différents groupes d' items, ce qui peut rendre plus difficile une sélection pour un utilisateur peu habitué à la présentation des données.

Impression.

Il est important de constater que l' impression peut se faire de différentes façons:

- utilisation d' une touche ("F5") programmée pour l' impression,
- utilisation de la touche "print screen".

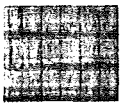
La première solution permet d' imprimer suivant des dimensions plus grandes mais sans voir ce qu' on produit.

La seconde solution permet d' imprimer ce qu' on souhaite mais sous la taille de l' écran.

Exemple:

Le document à imprimer dans le cas de l' horaire d' un professeur est la grille et se présente de cette façon:

(1)Il est bon de regarder "vocabulaire" dans 5.1.2.

	1	2	3	4	5	6	7	8
lun								
mar				1LaA math A102	1LaC math A402			
mer			1LaA math A203					
jeu				3MA math A104				
ven								

légende:

Dans ce cas nous nous contenterons des contraintes sur l' entité désirée.

 = heure indisponible pour le professeur

5.2.Autres solutions.

Tout ce que nous venons de voir jusqu' à présent relève d' un choix parmi d' autres.

Il ne s' agit pas de refaire l' application autrement mais de lancer des idées à développer éventuellement.

Ces idées portent surtout sur le traitement; en effet, l' ergonomie a été examinée plus profondément que le traitement.

On pourrait, non pas partir de données tout à fait indépendantes les unes des autres (sauf section-étudiant(e)), mais de données où des chemins existeraient entre toutes les entités, et se servir de ce fichier comme fichier de données.

Nous avons travaillé de la manière suivante:

Nous sommes partis de données indépendantes (obligatoire/option, local, professeur, local) et de deux données liées (section, étudiant).

La réalisation de l' horaire était de lier toutes les données indépendantes entre elles et de lier les données liées au données indépendantes.

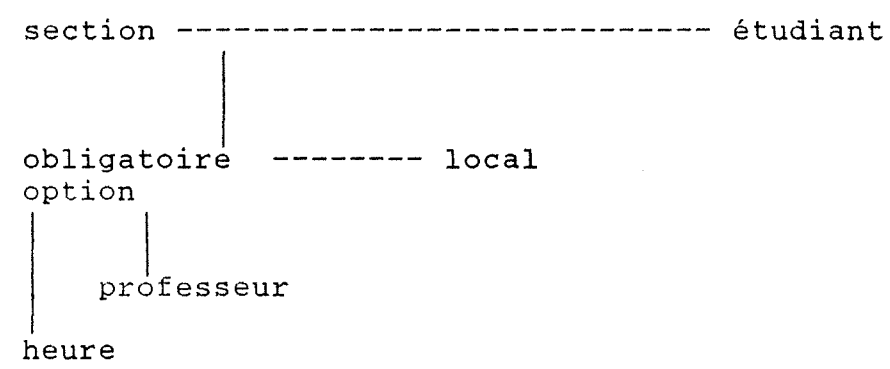
init:

Il y a initial ement 6 fichiers:

section ---------- étudiant

obligatoire local professeur heure
option

terminaison:



Le gros avantage de cette solution réside dans la facilité d' installation des données et surtout lors du traitement d' une très grande dynamique.

En effet, en cas de changement d' affectation d' une section pour un professeur, l' utilisateur n' a aucune difficulté.

Ce changement ne se passe vraisemblablement pas ou peu pour le secondaire supérieure mais il n' en va pas de même pour le secondaire inférieure ou les enseignants sont jeunes et ont généralement des morceaux d' horaires dans de nombreux établissements.

Une autre façon de travailler consisterait à partir de données liées dès le départ:

Dans ce cas, le seul lien à créer serait entre le groupe d' articles et l' article "heure".

En effet:

init:

Il y a initialement 2 fichiers:

```
section -------- obligatoire ---- local
étudiant           option


                heure
```

terminaison:

En fin d' exercice, on se retrouve avec un seul fichier rassemblant les deux fichiers de départ:

```
section -------- obligatoire ---- local
étudiant           option
                     |
                     |
                   heure
```

Cette solution permet une grande rapidité de traitement à partir du moment où les liens initiaux sont corrects.

Le changement d' affectation proposé ci-dessus poserait un problème et devrait se faire avant le travail de réalisation de l' horaire.

La question à laquelle il faudrait apporter une réponse avant de savoir quel solution appliquer est de déterminer de manière (temps) chiffrée les deux traitements suivant les deux solutions.

Solution1: (pas de lien au départ)

pré-traitement	traitement de l' horaire
.entrer les données de manière séquentielle	.créer tous les chemins entre les différents fichiers

Solution2: (liens au départ)

pré-traitement	traitement de l' horaire
.entrer les données liées .entrer les données heures	.créer un chemin entre les données liées et heure

La réponse à cette grille sera décisive, mais une réponse demanderait une étude détaillée.

Il nous semble en effet que cette réponse varie avec l' établissement.

Il est raisonnable de penser qu' un enseignant, au début de sa carrière, se trouvera confronté à des écoliers de n' importe quelle orientation, alors qu' en fin de carrière, il est raisonnable de penser que l' enseignant se situera dans une orientation où son cours est important.

Il est utile dès lors d' étudier la population des enseignants d' un point de vue statistique.

Un autre aspect pouvant influer sur les mouvances du corps professoral est peut-être également le type d' enseignement de l' établissement.

Le corps professoral d' un établissement professionnel peut, en tout cas dans ses cours généraux, ressentir certaines difficultés à conserver ses enseignants.

D' autres facteurs peuvent certainement encore intervenir dans la mouvance des enseignants, notamment des facteurs naturels qui sont la mise à la retraite et l' engagement de jeunes enseignants (les uns ne remplaçant évidemment pas les autres du point de vue de l' horaire).

On peut se demander si un horaire de cours ne pourrait pas s' utiliser dans d' autres **contextes considérés souvent comme très proches.**

Ce genre d' analogie est très DANGEREUX.

En effet, si par exemple on considère le cas d' un horaire d' examens, une étude attentive de l' analyse faite au chapitre3, conduit à remarquer que:

- la période de travail est différente: on parle de semaines pour les cours et de session pour les examens;

  l' interface de la grille dans ces deux cas est très différente,

- l' unité de travail est différente: on ne parle plus d' heure mais de demi journées,

- le schéma E/A est différent: les cours à option et les cours obligatoires existent encore, mais il faut distinguer en outre les examens oraux et les examens écrits.

- un professeur peut répartir une classe en plusieurs locaux pour faire passer des examens, il n' y a donc plus de restriction sur la taille d' une classe pour autant qu' il y ait du personnel surveillant.

- ...

On voit que le sentiment de similitude initial est remis en question.

Par contre, dans des **contextes tout à fait différents**, après étude de l' analyse, on pourrait conclure à une symétrie des différents problèmes.

Comment opérer?

Il suffit d' étudier:

- la partie ergonomique,
- la partie traitement.

Deux problèmes peuvent utiliser une même résolution non instanciée si tous les paramètres de l' analyse fonctionnelle et les primitives d' accès à la B.D. se correspondent.

Il faut,en tout cas, se méfier des conclusions hâtives.

5.3.Bibliographie.

[5.1.] D. SCAPIN

Guide Ergonomique des Interfaces Homme - Ordinateur

rapport INRIA - 1977

[5.2.] M.-F. BARTTHET

LOGICIELS INTERACTIFS et ERGONOMIQUES
Modèles et Méthodes de Conception

Dunod Informatique - 1988

[5.3.] B. SHNEIDERMAN

DESIGNING the USER INTERFACE
Strategies for Effective Human - Computer Interaction

Addison - Wesley Publishing Company - 1987

CONCLUSIONS

6.1.Conclusion.

Après une analyse très longue d' un problème qui, pour être très banal, n' est pas simple pour autant, nous arrivons à une solution qui nous paraît assez complète ou même, trop complète suivant la stabilité ou non du corps professoral de l' établissement scolaire.

En effet, nous avons développé une solution qui est assez lourde à utiliser car à chaque manipulation, nous devons entrer toutes les données et puis les lier.

Par contre cette solution présente l' avantage de permettre des différences d' affectation d' une année à l' autre ou même en cours d' année sans trop de difficultés.

Une autre solution aurait consisté a voir toutes les entités liées sauf "heure" et à n' avoir plus qu' à créer un chemin entre ce groupe d' entités et l' entité "heure".

Cette méthode est beaucoup plus facile à utiliser tant qu' il y a peu de changement dans le personnel enseignant, mais présente une grande rigidité en ce qui concerne notamment le changement d' affectation d' un professeur.

Prenons l' exemple d' un établissement où un professeur change d' affectation.

Dans la première solution, il suffit de faire appel à la fonction mofification et d' introduire de nouvelles valeurs:

```
professeur : AA
obligatoire: histoire-2
section    : 1LaA
local      : A302
heure      : 3eme heure du jeudi
```

Dans la deuxième solution, il serait demandé à l' utilisateur de changer de logiciel et de se servir de la partie installation des données (qui est très simple dans le premier cas et qui consiste à introduire les donnéees séquentiellement) et d' introduire à nouveau les données liées (ce qui n' est plus aussi simple car il faut introduire les données sous une suite de caractères qui répondent à un format déterminé).

Prenons l' exemple de la modification.

Nous avons créé un élément d' horaire comme suit:

```
professeur : AA
obligatoire: histoire-2
section    : 1LaA
local      : A302
heure      : 3eme heure du jeudi
```

Par la suite, nous constatons qu' il y a un chevauchement d' heure (par ex: ce professeur donne cours à une autre section à cette heure).

Nous souhaitons donc effectuer un changement d' heure.

Dans le premier cas qui est notre solution, il faut introduire:

professeur : AA
obligatoire: histoire-2
section : 1LaA
local : A302

A ce moment, on peut introduire une nouvelle heure.

On constate que 1' utilisateur a eu 5 opérations à effectuer.

Dans la deuxième solution, il aurait seulement suffi de définir:

valeur du groupe d' entités
heure

On se serait limité à 2 opérations.

Pour pouvoir faire un choix rationnel, il faudrait quantifier le temps total passé dans les deux méthodes.

Par temps total, nous entendons le temps avant traitement (ex: entrée des donées) et le temps de traitement.

Cette quantification semble très difficile à réaliser et surtout non généralisable. (chap5.2)

Certains privilégient la rapidité d' exécution de 1' application et d' autres la facilité d' utilisation.

Nous avons estimé que les utilisateurs revenaient chaque année au même point, c' est-à-dire à 1' état novice.

Pour cette raison, nous avons insisté sur la partie ergonomie.

En ce qui concerne 1' implémentation, nous avons passé beaucoup de temps sur les écrans, les menus déroulants, ..., et peu de temps sur le traitement sachant que de nombreux travaux ont été réalisés dans ce dernier domaine.

Ce programme compte en tout à 208 Kb dont 56 Kb ont été fournis par les F.N.D.P.

Niveau de Réduction de 1' Implémentation: (Annexe 1)

Outre le fait que les contraintes ne sont pas implémentées, le programme complet ne tourne pas car il présente un trop grand nombre de variables (plus de 64 Kb).

Individuellement, les routines fonctionnent mais 1' intégration semble difficilement réalisable sous Turbo-Pascal3.

Il semble nécessaire de passer sous une forme plus performante de Turbo-Pascal (4 ou 5).

La raison pour laquelle ce travail a été réalisé en Turbo-Pascal3 est dû au fait que, initialement, la base de données NDBS était prévue pour tourner sur ce pascal.

L' intégration comporte deux parties:

- intégrer l' ergonomie (réalisé),
- intégrer les primitives de la B.D. (non réalisé).

Les primitives d' accès à la B.D. sont réalisées mais non intégrées; ce qui présente l' inconvénient que aucun traitement ne peut être effectué, à la rigueur très peu de consultation.

Par contre, tous les fichiers peuvent être considérés comme des outils à une réalisation prochaine d' un horaire de cours, moyennant modifications, en Turbo-Pascal4.

Vu le grand nombre d' information échangé avec l' utilisateur, il est conseillé d' utiliser un disque dur.

Outre l' aspect technique, l' aspect conceptuel mérite qu' on s' y attarde et qu' on tâche de répondre à un certain nombre de questions.

Certaines questions restent en suspend et sont effleurées dans 5.2.

ANNEXES

ANNEXE 1

Nous reprenons dans cette annexe l' ensemble des fichiers implémentés pour la réalisation de ce logiciel.

Nous avons divisé notre programme en fichiers:

- essai_co.pas: programme principal a pour but de lier toutes les procédures et tous les fichiers (55 Kb)

- constant.pas: fichier reprenant les constantes de ce programme (2.8 Kb)

- var.pas: fichier reprenant un grand nombre de variables globales du programme (487 b)

- typeswin.inc: fichiers comprenant les types d' objets nécessaires au travail sur les fenêtres (712 b)

- message2.pas: fichier permettant la lecture des messages mis dans un fichier séquentiel (379 b)

- p_profess.pas: fichier responsable du titre des écrans de saisie de données, des déplacements dans ces écrans et des fonctions utilisables au bas de chaque écran (2.5 Kb)

- ecran1.pas: fichier responsable des tracés de grilles horaire, des déplacements dans ces grilles, de la reconnaissance d' un point dans la grille (8 Kb)

- menu.pas: fichier responsable des sélections dans le menu déroulant (vertical et horizontal) et de la définition des fenêtres qui pourront être sauvées en mémoire d' écran par après (15 Kb)

- l_com: fichier permettant de lister les entités cours obligatoire (401 b)

- l_etud: fichier permettant de lister les entités étudiant (217 b)

- l_loc: fichier permettant de lister les entités local (242 b)

- l_pr: fichier permettant de lister les entités professeur (220 b)

- l_sect: fichier permettant de lister les entités section (344 b)

- l_op: fichier permettant de lister les entités option (216 b)

- bd.pas: fichier permettant les créations de chemins, les suppressions de chemins, les scannes de chemins (45 Kb)

- primitiv.lib: fichier créant un librairie de primitives telles que tout ce qui touche au travail dans des fen@tres, au sauvetage et rechargeagement des fen@tres (11.5 Kb)

- routines.lib: fichier créant une autre librairie de primitives portant également sur le travail dans les fen@tres (4.3 Kb)

- horaire.typ: fichier généré par le dictionnaire de la B.D., reprend l' ensemble des types des articles de la B.D. (5 Kb)

- dbms: fichier fournissant les primitives élémentaires de la B.D. qui seront utilisées sous forme de macro-primitives (56 Kb)

Ce programme compte en tout à 208 Kb dont 56 Kb ont été fournis par les F.N.D.P.

ANNEXE 2

RAPPORT SUR UNE BASE DE DONNEES

****

NOM: HORAIRE
Taille Buffer: 2 à 30 K: 30
Description:

Liste des types d'Entités
==========

1 : SECTION
    Mode Stockage: C
    Description:
    Attributs:
      NUMSEC : String[4]
      TAILLESEC : Integer

    Description Attribut:
      NUMSEC :
      TAILLESEC :

2 : ETUDIANT
    Mode Stockage: C
    Description:
    Attributs:
      NUMET : Integer
      NOMET : String[30]
      PRENET : String[30]

    Description Attribut:
      NUMET :
      NOMET :
      PRENET :

3 : SECT_OBL
    Mode Stockage: C
    Description:
    Attributs:

    Description Attribut:

4 : ET_OP
    Mode Stockage: C
    Description:
    Attributs:

    Description Attribut:

5 : OBLIGATOIRE
    Mode Stockage: C
    Description:
    Attributs:
      NUMOB : Integer
      NOMOB : String[20]

EX_CONT_LOC_OB : Char
CONT_LOC_OB : array [1..5] of String[4]
HEURE_OB : Integer

Description Attribut:
NUMOB :
NOMOB :
EX_CONT_LOC_OB :
CONT_LOC_OB :
HEURE_OB :

6 : OPTION
Mode Stockage: C
Description:
Attributs:
NUMOP : Integer
NOMOP : String[20]
EX_CONT_LOC : Char
CONT_LOC : array [1..5] of String[4]
HEURE_OP : Integer

Description Attribut:
NUMOP :
NOMOP :
EX_CONT_LOC :
CONT_LOC :
ensemble des locaux utilisables pour ce cours s' il est
is à une telle contrainte
HEURE_OP :
nombre d' heure de cours

7 : OBL_PR
Mode Stockage: C
Description:
Attributs:

Description Attribut:

8 : OP_PR
Mode Stockage: C
Description:
Attributs:

Description Attribut:

9 : PROFESSEUR
Mode Stockage: C
Description:
Attributs:
NUMPR : Integer
NOMPR : String[30]
PRENPR : String[30]
EX_CONT_IND_PR : Char
HEURE_IND_PR : array [1..10] of Integer

Description Attribut:
NUMPR :
NOMPR :
PRENPR :
EX_CONT_IND_PR :
HEURE_IND_PR :

10 : OP_LOC
Mode Stockage: C
Description:
Attributs:

Description Attribut:

11 : OB_LOC
Mode Stockage: C
Description:
Attributs:

Description Attribut:

12 : LOCAL
Mode Stockage: C
Description:
Attributs:
NUM_LOC : String[4]
EX_CONT_ACT : Char
CONT_ACT : Integer
EX_IND_LOC : Char
IND_LOC : array [1..5] of Integer
TAILLE_LOC : Integer

Description Attribut:
NUM_LOC :
EX_CONT_ACT :
CONT_ACT :
EX_IND_LOC :
IND_LOC :
TAILLE_LOC :

13 : OBL_HEURE
Mode Stockage: C
Description:
Attributs:

Description Attribut:

14 : OP_HEURE
Mode Stockage: C
Description:
Attributs:

Description Attribut:

15 : HEURE
Mode Stockage: C
Description:
Attributs:
NUMHEURE : Integer

Description Attribut:
NUMHEURE :

Liste des Types d'Association
==========

1 : A1

Origine: SECTION
Cible: ETUDIANT
Description:

2 : B1

Origine: SECTION
Cible: SECT_OBL
Description:

3 : B2

Origine: OBLIGATOIRE
Cible: SECT_OBL
Description:

4 : B3

Origine: OBLIGATOIRE
Cible: OBL_HEURE
Description:

5 : B4

Origine: OBLIGATOIRE
Cible: OB_LOC
Description:

6 : B5

Origine: OBLIGATOIRE
Cible: OBL_PR
Description:

7 : B6

Origine: PROFESSEUR
Cible: OBL_PR
Description:

8 : B7

Origine: LOCAL
Cible: OB_LOC
Description:

9 : B8

Origine: HEURE
Cible: OBL_HEURE
Description:

10 : C1

Origine: ETUDIANT
Cible: ET_OP
Description:

11 : C2

Origine: OPTION
Cible: ET_OP
Description:

12 : C3

Origine: OPTION
Cible: OP_HEURE
Description:

13 : C4

Origine: OPTION
Cible: OP_LOC

Description:

14 : C5

Origine: OPTION
Cible: OP_PR
Description:

15 : C6

Origine: PROFESSEUR
Cible: OP_PR
Description:

16 : C7

Origine: LOCAL
Cible: OP_LOC
Description:

17 : C8

Origine: HEURE
Cible: OP_HEURE
Description: